The VLSI Design Of A Simple-Instruction 16-Bit Microprocessor

By

Joseph E. Varrientos

BSEE, Kansas State University, 1986

A MASTER'S THESIS

submitted in partial fulfillment

of the requirements for the degree

MASTER OF SCIENCE

Department Of Electrical And Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1989

Approved by:

Andrej Rys

Major Professor

TABLE OF CONTENTS

List Of Figures

List Of Tables

Acknowledgments

The advise, guidance, and support of many individuals went into the completion of this project. All of them were necessary for this chip to work, and I would like to thank them here.

First, I would like to thank Phillip Buckland. His commitment to the support of the VIVID CAD tools was essential to the success of this project. His patience, tolerance, and experience were greatly appreciated.

In addition, I would like to thank my major professor Dr. Andrzej Rys for his guidance and support. Dr. Rys gave me the freedom I needed to explore all the avenues in VLSI design I wished. I greatly admire the effort Dr. Rys put out for me in getting the tools and equipment I needed, and I will always be at his service.

But most of all, I would like to thank my father Eugene C. Varrientos, for his faith and support of my efforts over the past eight years. He was the only force I could depend on when the going got really tough. He kept me from financial ruin, pulled me through times of disillusionment, and gave me the strength to believe in myself and by abilities. Thanks, Dad.

VLSI Design Of A Simple 16-Bit Microprocessor

1.0 Introduction

For this thesis, the VLSI design of a simple 16-bit microprocessor was constructed. This machine is an upgraded version of the 8-bit machine discussed in Devore and Hardin[1]. This 16-bit version was called the TORO 680-16. Its construction was an exercise in VLSI circuit design to demonstrate the relative power of the design tools obtained by and the computing power available in the Department of Electrical & Computer Engineering at Kansas State University.

This thesis begins with a discussion of the design tools and design methodology used in constructing this design. A justification for the integrity of the standard cell library constructed and used for the microprocessor is also given. The discussion follows with a description of how the register set, arithmetic logic unit, program counter, control logic, and the I/O pads were designed and constructed. Description of the simulations used to check each of the above designs for functionality and performance is also given. The discussion continues with the simulations for the final design and how those simulations were performed. The results for pertinent

signals from the simulations are summarized and the final system timing is calculated. For completeness, a discussion is given for possible future improvements and implementations. In addition, three appendices are included with information on design simulation and construction.

1.1 Design Tools

The CAD layout tools used in constructing this design were obtained from the Northwest Laboratory For Integrated Systems at the University of Washington[2], and the Microelectronics Center Of North Carolina (MCNC)[3]. The computer systems used were a SUN 3/60 running Berkeley UNIX 4.2 release 3.5, and a Digital Vax 11/750 running Wollongong Eunice BSD 4.3. Layout editors in both systems of tools were used, but for different stages of construction. In addition, SPICE 3A7 was used to characterize the pads and three-state buffers constructed, and simulation tools in the package from MCNC were used to verify final design inter-connectivity and functionality.

The design rules and SPICE parameters used were available from the fabrication foundry MOSIS at the University of Southern California[4], in anticipation of its use as a fabrication foundry for the design. The design was constructed using the 3-micron, bulk p-well

scalable CMOS process technology, and was inserted into the MOSIS standard 6800 micron by 6900 micron pad frame. Final transistor count including the pads was 14,156.

It had been hoped early in this design effort that the "hands-off" mask layer generator available in the VIVID CAD package from MCNC would be sufficient in creating the mask layers for this layout, but this was not the case. At first, it was found that the compacting algorithms used by the VIVID tool HCOMPACT created stretched cells that were geometrically acceptable, that is, with small drain and source regions. However, as the design grew, this stretching became more pronounced, due to the pitch-matching HCOMPACT attempts to do for hierarchical designs, and the growing amount of irregularity in the control logic portion of the design. The stretched cells soon became too long to be acceptable, because of the excessive resistances and capacitances created at drain and source regions. Thus, the VIVID package was used to construct the standard cell library for the design, and to generate mask layers for the control logic functional block alone. The VIVID simulator FACTS was used to simulate the various functional blocks. MAGIC, the layout editor from UC-Berkeley, was used to construct macros and assemble the final design.

1.2 Design Methodology

First, the standard cell library constructed for the TORO was created in the VIVID interactive editor ICE. The construction of these cells was somewhat modeled after the cells available in the CMOS3 Library[5] and Volume 3 of the VIVID version 1.3 Designer Documentation[6], specifically, the Standard Circuit Module Library. The cells constructed were then converted from their symbolic representation in the A Better Circuit Description (ABCD) language[7] into a VIVID internal layout language called LLAMA by the VIVID tool HCOMPACT. Another tool from VIVID, called ATOLL, was used to translate the VIVID internal layout language to the California Intermediate Form (CIF). Then, using MAGIC, the cells were again "standardized", because the HCOMPACT compaction process created cells with differing heights. Thus, two libraries of cells were supported: one for VIVID and one for MAGIC. However, only the MAGIC cells appear in the final layout. Labels were also attached. Plots and transistor schematics for the MAGIC cells are given in Appendix A.

Once the standard cell library had been established, the design proceeded as follows: first, a macro was constructed in VIVID ICE and simulated using VIVID FACTS to show functionality and design integrity. Once this was

shown, the macro was then constructed using the MAGIC editor. Thus, there were two layouts supported, and only the VIVID layout could be checked by simulation. Currently, the software needed to use the extracted circuit parameters from the MAGIC editor have not been successfully installed on the SUN workstation used for this design. Thus, signal tracing of the final composite mask layout was the method by which the final CIF design was checked for proper inter-connectivity. Plots for the final layout were provided by Glen Hush of Micron Technology, Inc., Boise, Idaho. The signal tracing proved to be a large task, but the hierarchical construction and relatively small size of the final layout provided some reduction in the complexity of this task.

Another early consideration was in design philosophy. It was necessary to decide whether to design additional standard cells for two-phase timing, include PLAs, and to decide what busing scheme would be used. The following decisions were made based on the experience and resources available at the time:

1) The design was laid out in "silicon compiler" fashion, that is, as long lines of discrete logic gates inter-connected above and below as needed,

with a common Vdd and Vss bus. This decision led to the creation of the standardized cell library.

2) Two-phased timing was not used. Logic cells were constructed based on Euler's method described by Weste[8], and a single phase clock was incorporated. All registers were edge-triggered D-type flip flops. Additional inputs were added to the flip-flops to allow a loading feature.

These conventions made it more simple to understand the system at the logic level, and allowed system timing requirements to be less strict. This also allowed for a smaller cell library, and dictated the use of passive buses.

3) Control logic was implemented by inter-connecting discrete logic. This decision was made because of the editors available at the time, and because early indications proved that the PLA implementation would be quite slow, given the large number of minterms in some control logic equations.

Another early consideration in the TORO layout was in floorplanning. The TORO design required a long internal common data bus to accomplish the transfer of register

data. It was decided that as much of the data flow logic as possible would be laid out on a continuous power bus, and that three-state logic would be made powerful enough to handle the excessive capacitance. Indeed, the three-state driver cell succumbed to several design iterations. When the layout neared completion, it was only possible to allow the register set and ALU to remain on a continuous bus; the program counter was "folded" and designed to have common power bus with the control logic functional block. A figure showing the locations of the four large functional blocks in the layout appears in Figure 1.

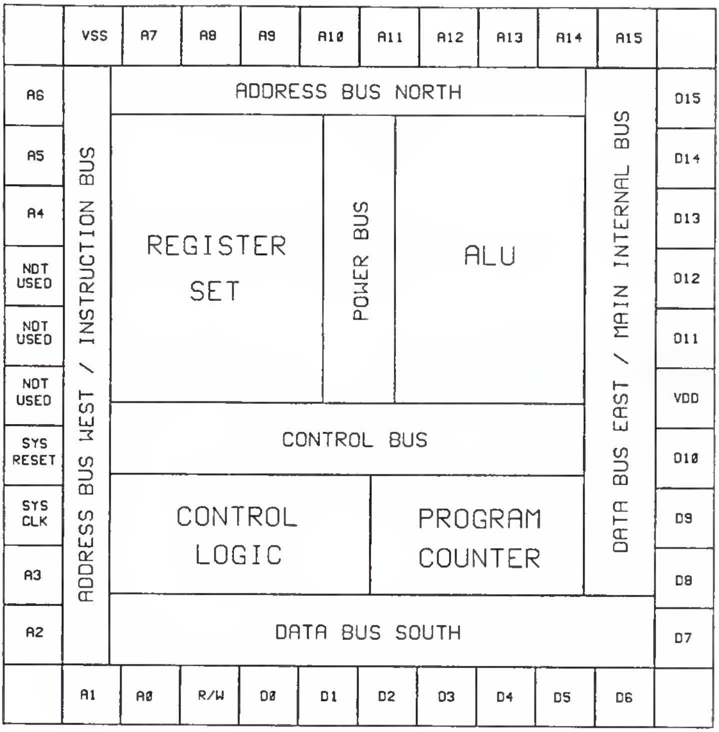

Figure 1: Layout Of TORO680-16

The floorplanning scheme also included a convention for control signal propagation. All control signals for the data flow logic were made available at the bottom of each large functional block, and allowed to propagate, when appropriate, through standard cells to other cells and/or macros above. The majority of the inter-connect material for these control signals paths was poly-silicon, so some concern was expressed early as to design rule violations involving long wires of polysilicon, because of the linear voltage drop along these signal paths due to finite resistance. However, the final design meets all the design rules allowed by the MOSIS 3-micron process. In the control logic and program counter sections, the above convention was not used, given the large amount of irregularity in those two functional blocks. However, all other long paths for data and control signals for the control logic and the data flow logic was routed on metall and/or metal2.

In determining the I/O pad placement, the overlapping of signal buses was considered. For the final layout, there were four buses to consider:

1) The Main Internal Bus connecting the ALU/Register Set with the Program Counter,

2) The Instruction Register Bus connecting the instruction register with the Control Logic,

3) The Address Bus connecting the Register Set with the address I/O pads,

4) The Data Bus connecting the Main Internal Bus to the data I/O pads.

The main internal bus was, for most purposes, the data bus with some three-stating for data flow control. Thus, those two buses were placed on the same side of the TORO. The address and instruction register buses were allowed to occupy the other side. The pads for the read/write, the system clock, and system reset were interspersed among the address bus so that they could be physically close to those signal inputs. Figure 1 shows the pad placements relative to the TORO functional blocks.

1.3 The Eight-Bit "Tiny Chip" Counter

Characterization by simulation for the two standard cell libraries was not performed. For the MAGIC library, this characterization was impossible due to the lack of properly installed software. The VIVID library, however, was not characterized because the effort required much repetition and computer time. Because of time constraints,it was decided that characterization of the

VIVID standard cell library would be adequately accomplished by the fabrication of a "tiny chip". Thus, the eight-bit counter was laid out and assembled into the MOSIS 2300 micron by 3400 micron standard frame using MAGIG and sent for fabrication at MOSIS.

The eight-bit counter was constructed from two modified macros used for the program counter described in section 2.4. Modifications were made so that d-flip flop data inputs could be observed as well as their q outputs. These additional observation points allow the chip to be more fully at characterized. A report is currently being completed that gives results from computer simulation using the VIVID tool FACTS for worst case circuit parameters. Upon the chip's return from MOSIS, the chip will fully characterize, giving an indication of the accuracy of the simulation and a measure of the standard cell library integrity. This body of work should be completed by December, 1989. A block diagram layout of the tiny chip appears in Figure 2.

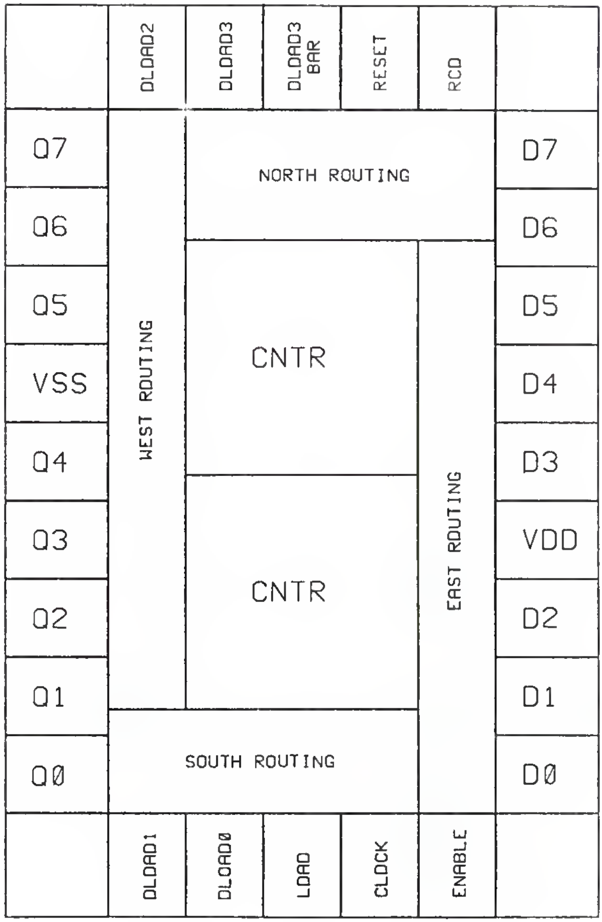

Figure 2: Layout of Eight-Bit "Tiny-Chip" Counter

2.0 The TORO 680-16

The TORO 680-16 is a 16-bit microprocessor with four addressing modes and twenty-six instructions. Its behavioral specifications exactly match those of the TORO machine described by Devore[9], but has been modified in three important ways:

1) All data flow was increased from eight bits to sixteen bits to increase the size of the processor's memory map space.

2) Additional read/write control circuitry was added to accomplish I/O pad three-stating and a read/write hardware output.

3) A write data register was added to accommodate write timing for external memory.

A complete description of the machine, register transfer information, and instruction assembly is well summarized in Devore, and excerpts from that paper including the system's block diagram, register-transfer information, control signal equations and other supporting information appear in Appendix C.

The location of signal pin-outs for the die is shown in Figure 1. Note that only three pins are used for system control. For this application, no other control pins were

needed or required. Three pins in the 40-pin pad frame were unused and made available for future circuit modification or testing.

2.1 TORO 680/16 Layout And Construction

In laying out the TORO, the conceptual design provided by Devore[10] was "filled in", that is to say, the ALU, program counter, and other control circuitry was designed at the logic level before layout was begun. Some re-design occurred during layout, but none that changed the original TORO behavioral specifications or system functionality. The TORO was sub-divided into four major functional blocks:

1) Register Set
2) ALU
3) Program Counter
4) Control Logic

The divisions were made along natural boundaries among the system's functional elements. These divisions proved to be advantageous in performing final system simulation, floorplanning, and layout verification. The system and its divisions are shown in Figure 3.

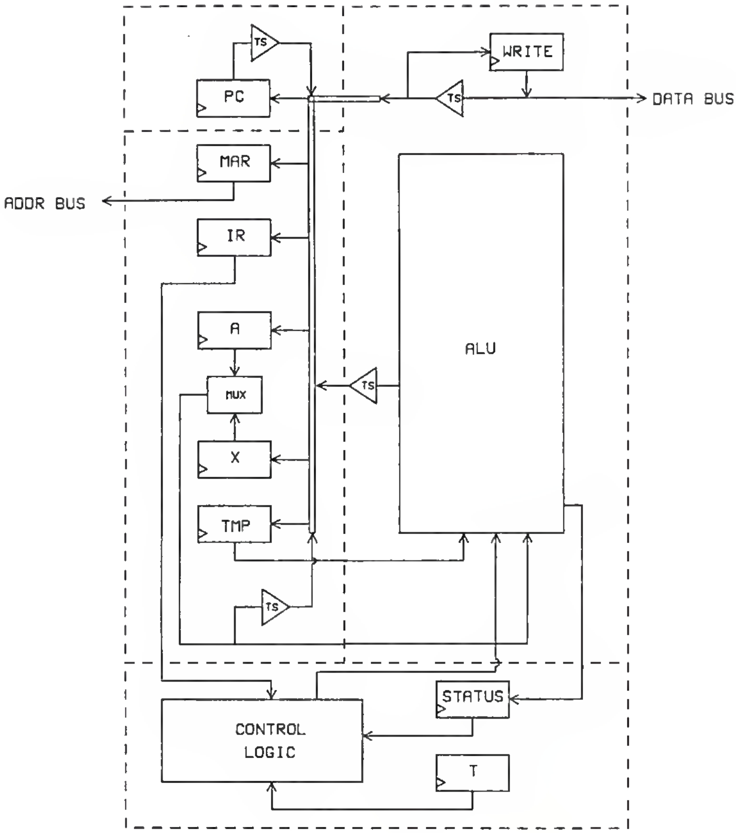

Figure 3: Block Diagram Of The TORO680-16

2.2 Register Set

The construction of the register set was the simplest of the four functional blocks. It was constructed using the Magic layout editor. The register set included the instruction register (IR), the memory address register (MAR), the temporary register (TMP), the accumulator (A), and the index register (X). Also included in this functional block are some discrete gates for control signal multiplexing, a two-to-one multiplexer for multiplexing the outputs of A and X, and three-state buffers for controlling access to the main internal bus by the A and X registers. Also worthy of note are the non-inverting buffers, NIV. They were included at the outputs of the MAR, the IR, the TMP and the output of the 2-to-1 multiplexer because the large load capacitances those cells were required to drive. A complete list of names for the standard cells appears in Appendix A.

A figure for one bit of the 16-bit register set appears in Figure 4. This figure shows the relative position of the cells used in this macro, and gives some signal input/output locations. Note that the bus labeled "mainbus" is the main internal bus, and that it runs the length of the macro. To the right of the register set, this bus connects to the ALU, and again runs the length of

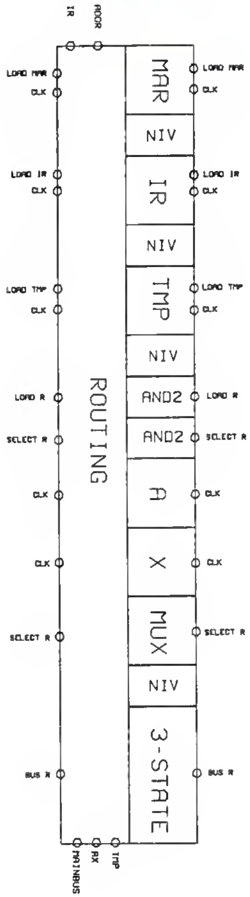

Figure 4: Layout Of One Bit Of The TORO680-16 Register Set

the routing found in that functional block. Also note that the control signals are sent to this macro from the bottom and are allowed to propagate upward. This macro was constructed such that any number of macros could be cascaded to give the register set needed. This cascading is shown in Figure 5, and shows the layout construction of the TORO register set.

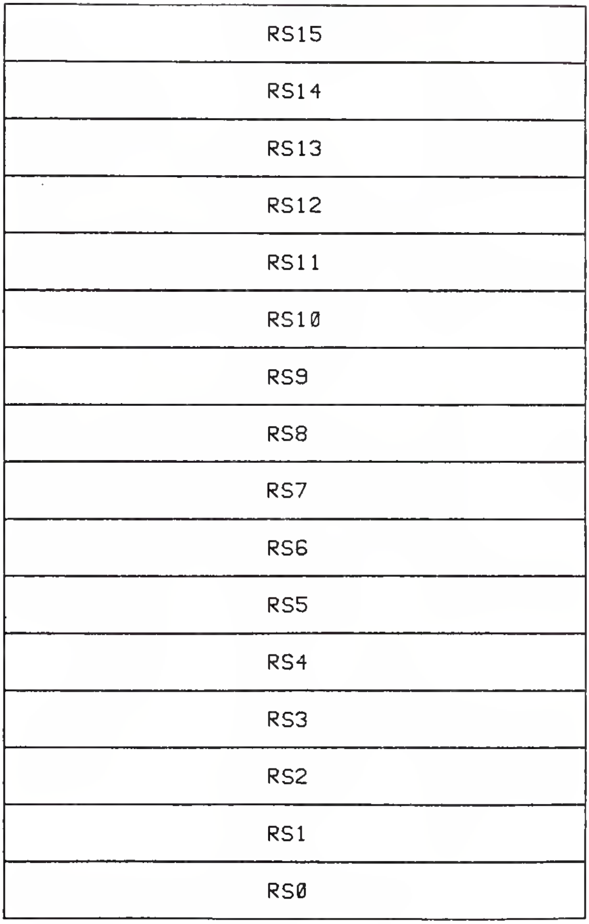

Figure 5: Layout Of The TORO680-16 Register Set

2.3 Arithmetic Logic Unit

In the design by Devore[11], the ALU was left to the student as a black box. After some review of Langdon[12], the ALU was designed around a four-bit carry look-ahead adder taken from Weste[13]. Thus, the ALU was more appropriately constructed from four-bit macros, rather than from one-bit macros, as was done for the register set. The ALU logic design appears in Figure 6.

From the figure, one may note that the majority of the multiplexing was accomplished with discrete gates. This multiplexing was used to give the needed "1" or "0" at the inputs of the exclusive OR and adder portions of the ALU. AND, OR, and exclusive OR operations were accomplished with discrete gates, then multiplexed through the adder and ALU output multiplexer. Shift and Roll operations were performed with the ALU output multiplexer. The Test, Compliment, Subtract, and Compare operations negate one operand using the exclusive OR, and are then added appropriately to the second operand to give the desired result and/or status bits. A complete list of instructions appears in Appendix C.

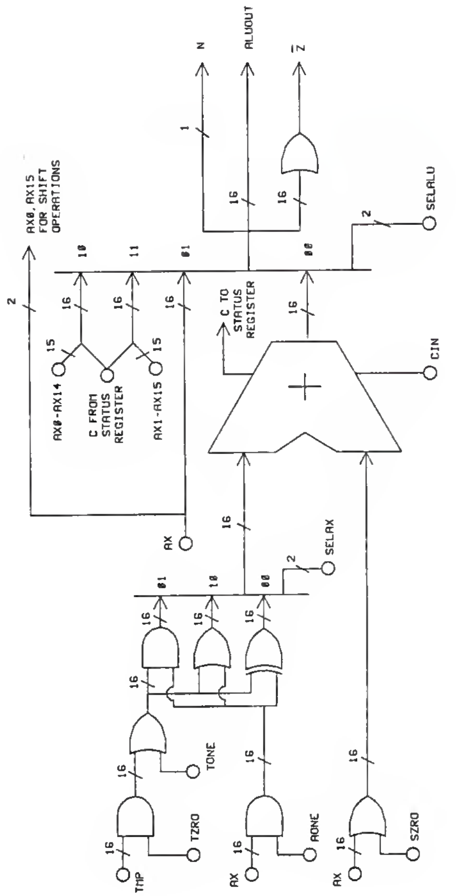

Figure 6: Logic Diagram Of TORO680-16 Arithmetic Logic Unit

In order to generate the ZERO bit for the status register, OR gates were cascaded together to create a 16-input OR gate. The inputs to the 16-input OR gate were the output of the ALU. The output of the OR gate generated the inverse of the ZERO signal that was saved by the status register, which is located in the control logic functional block. Recall that a ZERO status signal is generated when the output of the ALU is a zero. In Figure 7, the relative position of the functional portions of the ALU are given, along with bus locations. Again, this macro and functional block was constructed using MAGIC.

Note from Figure 7 that the write data register was included in the ALU macro. This register was included here to bring the register physically closer to the data I/O pads. The outputs of the register were routed directly to the OUT inputs of the I/O pads. Three-state cells were also included to isolate the IN pin of the I/O pads from the main bus and were enabled only during read operations. This is shown in more detail in Figure 8. The I/O pads are discussed in more detail in section 2.6.

For this large functional block, two versions of the four-bit macro were created. One is the cascadable version of the four-bit ALU, and the other is a modified version. This modified version has routing that is different than

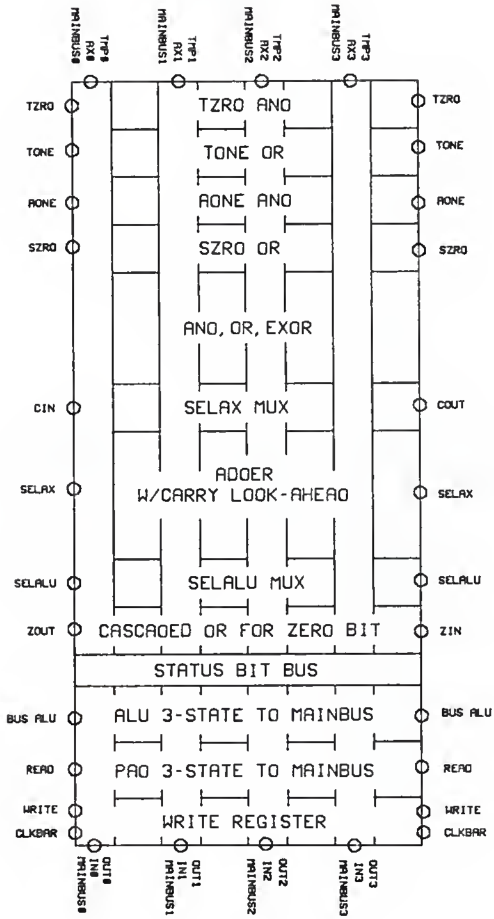

Figure 7: Layout Of Four Bits Of The TORO680-16
Arithmetic Logic Unit

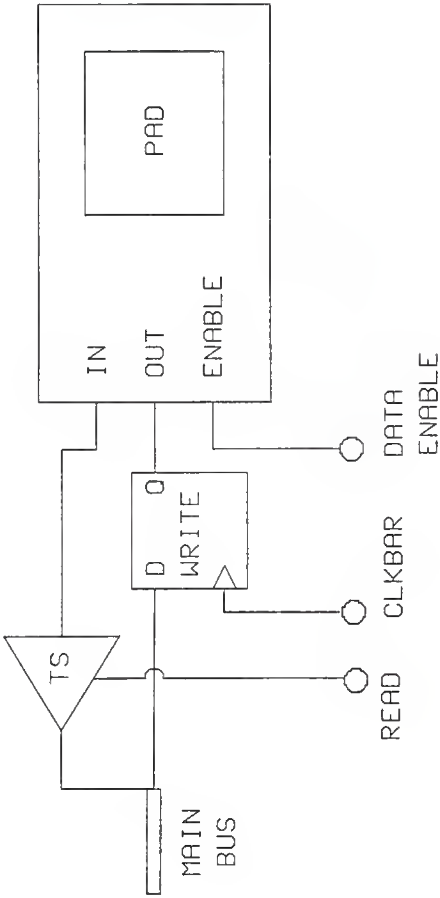

Figure 8: Logic Design Of Data Flow From Pads To TORO680-16 Main Internal Bus

the cascadable version, but only for the most significant bit. The additional routing propagates the output of the most significant bit, and the carry out bit, to the bottom of the ALU. These signals are stored by the status register for the NEGATIVE and CARRY bits, respectively. Other routing was included for the passing of the carry bit, stored in the status register, to the most significant bit of the ALU during Roll and Shift operations. Figure 9 shows the relative positions of the ALU macros in the TORO 680-16 Arithmetic Logic Unit.

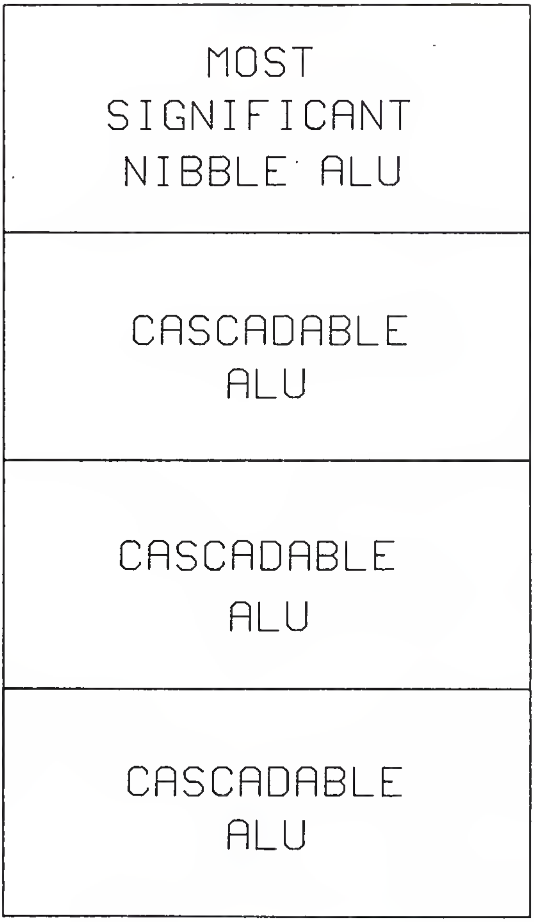

Figure 9: Layout of 16-Bit TORO680-16 Arithmetic Logic Unit

2.4 Program Counter

The program counter for the TORO is a 16-bit binary counter which features parallel load inputs, load and count enables, and a ripple carry out for circuit cascading. The program counter constructed was modeled from the schematic found in the Texas Instruments TTL Data Book for the 74LS163[14]. The boolean expressions used in constructing the counter is given in Appendix C.

Again, this functional block was constructed from four-bit macros using the MAGIC layout editor. Figure 10 shows the relative positions of the cells used in the counter macro. Note that the main internal bus connections exist at the right of the macro. Because of available silicon real estate, the program counter could not be floorplanned like the register set and ALU, that is, with continuous power buses for the entire 16 bits by the simple cascading of the macro. The program counter was "folded" into a block eight rows of cells wide. Additional routing was constructed so that the main internal bus could be made as short as possible. Thus, this program counter was customized to fit the given space. Figure 11 shows the locations of the counter macros in the program counter.

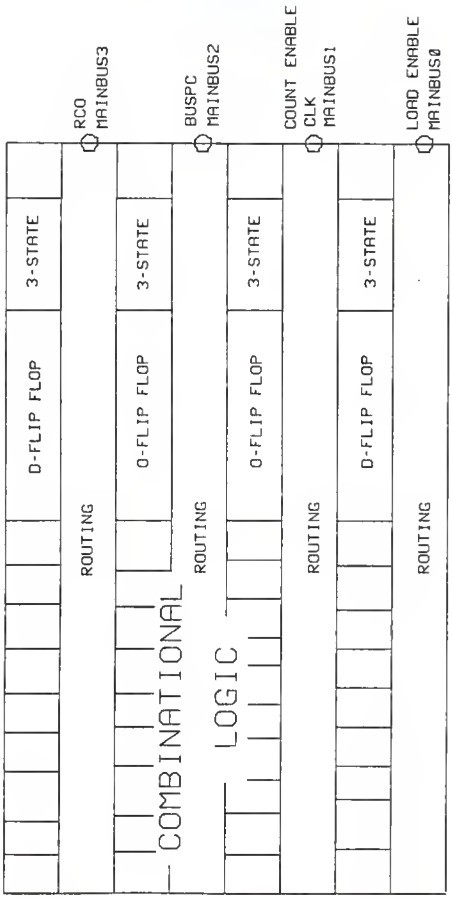

Figure 10: Layout Of Four Bits Of The TORO680-16 Program Counter

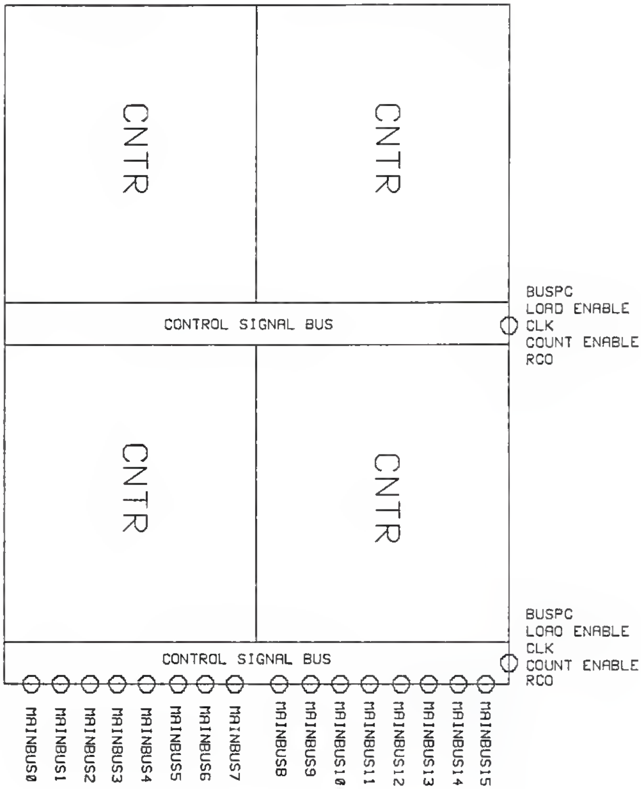

Figure 11: Layout Of The 16-Bit TORO680-16 Program Counter

2.5 Control Logic

The control logic was by far the most difficult functional block to construct. Included in this block, in addition to the control signal generation described by Devore[15], was the status register, the phase clock T, the control logic for ALU control signal generation, multiplexing for the carry bit, and additional logic for the read/write output signal and pad three-stating. The difficulty in constructing this block arose from the irregularity of the layout and the complexity of the inter-connectivity. Figure 12 shows the relative placement of cells groups which perform the various control functions. Much of the real estate is consumed in providing large drivers for the control signals and in providing buses for signal routing.

The composite mask layout for this functional block, unlike the other three, was constructed using the HCOMPACT tool from VIVID. It was mentioned earlier that HCOMPACT, when given a hierarchical design with much irregularity, would give stretched standard cells that in many cases were unacceptable. Because of the enormous task involved in inter-connecting the cells in this functional block using the VIVID editor ICE, however, it was not desired to repeat this effort in MAGIC. To circumvent this task,

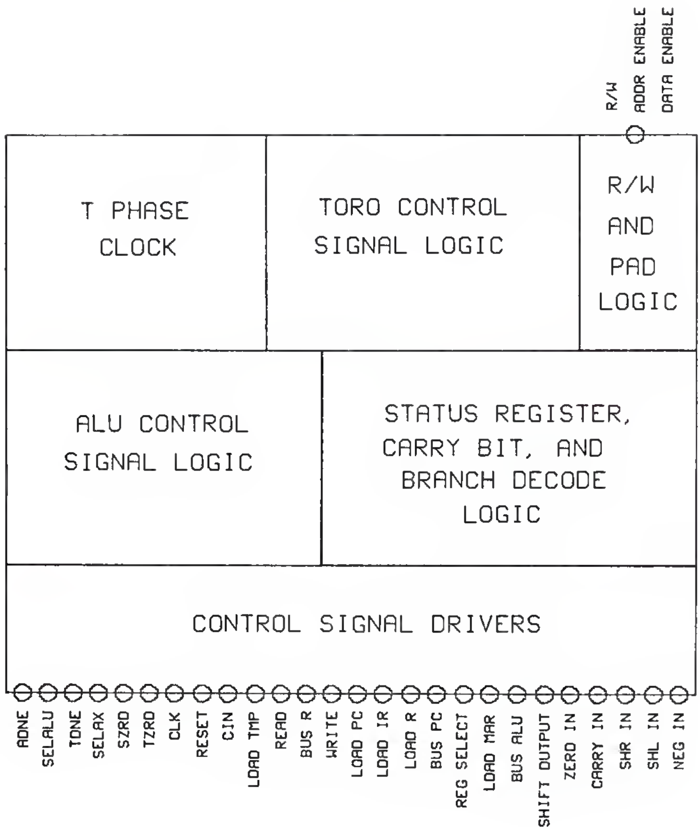

Figure 12: Layout Of The TORO680-16 Control Logic

HCOMPACT was used to generated the mask-level design. The resulting standard cells were then "hand-edited" using MAGIC to remove the stretching created. Although a moderately time-consuming task, this hand-editing of the standard cells was a less time-consuming and a more reliable method of obtaining a correctly inter-connected mask-level representation of the control logic functional layout than could have been accomplished by repeating the routing effort in MAGIC.

In Appendix C, the control signal equations for the TORO are given. Additional equations are given for the ALU control signals and for the T clock. Note that for the TORO control equations, some signals are a subset of the others. This subset of signals was used in generating the more complex control signals. This made some signals inherently slower than others, but simulation has shown that the differences in propagation delay time were small enough that all the signals were of the same relative magnitude.

A small bit of circuitry was added to the control logic to accomplish carry-bit input multiplexing. A logic diagram for this multiplexing is shown in Figure 13. For most instructions, the input to the carry bit is exactly that which comes from the ALU. For CMP, TST, and SUB

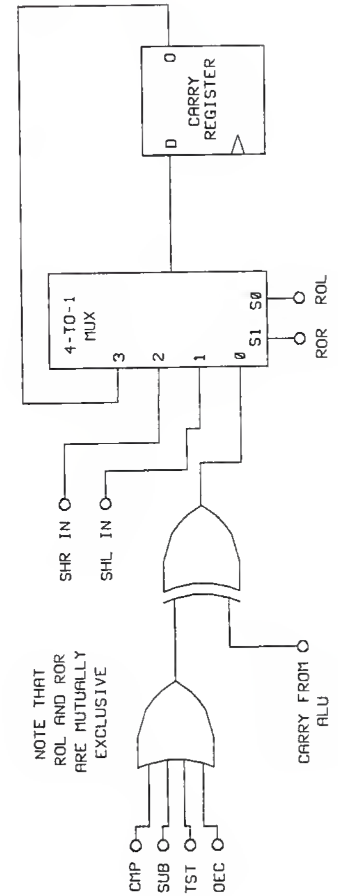

Figure 13: Logic Diagram Of The Status Register Carry Bit Decode Circuitry

instructions, the carry output from the ALU was inverted before recorded by the carry bit register. During ROR and ROL instructions, the carry bit register input was multiplexed appropriately from the outputs of the ALU. In addition, the carry bit register output was multiplexed via a single AND gate to the ALU output multiplexer during SHL, SHR, ROR, and ROL instructions.

The generation of the branch control signal was accomplished by cascading four-to-one and two-to-one multiplexers together to create an eight-to-one multiplexer. The inputs to the multiplexer were the appropriate outputs from the status register. Figure 14 shows the logical diagram for the multiplexing of the branch control signal.

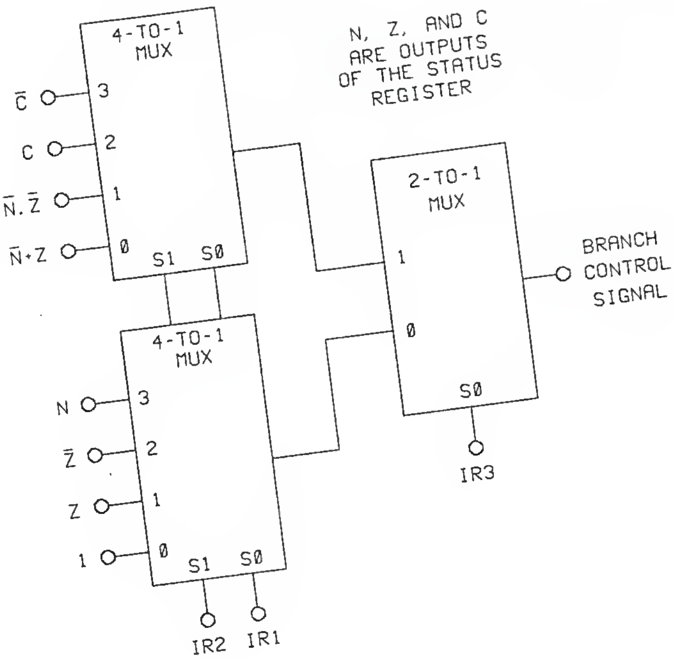

Figure 14: Logic Diagram Of The Branch Instruction Decode Circuitry

2.6 I/O Pad Construction

The I/O pads used in the TORO design were modified from the I/O pad set obtained from MOSIS. This pad set was designed at the Massachusetts Institute Of Technology for the MOSIS 3-micron SCMOS process. The pads were modified to increase their internal and external driving capability, and thus, their external static protection. This was done by increasing the size of the output pad and input driver transistors. It was anticipated that this design, once fabricated, would be used in the laboratory and handled by many students. The exact increase in static protection is not known; the figure given from MOSIS for the original pad was 3000 Volts. The increase in output transistor size from the original was about 33 percent. The current method for accurately testing this parameter is a destructive test after fabrication.

Figure 15 gives the logic diagram of the I/O pad. The power pads for Vdd and Vss are un-modified, except for additional power bus stretching, so that they could more easily fit into the MOSIS 6800 micron by 6900 micron pad frame. A more detailed transistor schematic of the I/O pad appears in Appendix A.

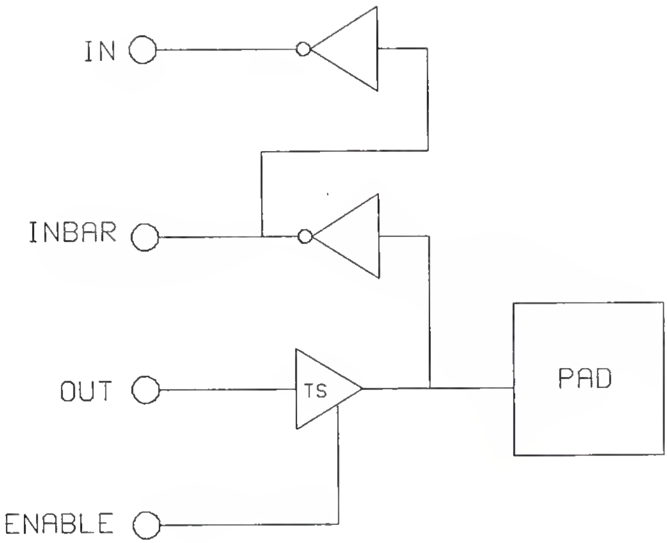

Figure 15: Logic Diagram Of The I/O Pad

2.7 Final Floorplanning

The MAGIC editor was used to assemble the pad frame and rout buses to the pad pins in the final composite mask layout. Because of the automated device bonding and packaging service provided by MOSIS, it was necessary to obtain from them a pad placement document giving the exact location for pads in the pad frame. The following file gives the pad locations for the MOSIS 6800 micron by 6900 micron pad frame. The coordinates given correspond to the center of the bonding pad in lambda units. To convert the coordinates to microns, multiply them by the minimum feature width. For this design, the minimum feature width was 1.5 micron per lambda.

```
*********************************************************
*                                                       *
*  This standard pad frame is for the TORO680-16 design *
*  The pad locations are for the 1.5 micron/lambda      *
*  feature width required by MOSIS for the              *
*  3.0 micron bulk p-well process they support.         *
*                                                       *
*********************************************************
*                                                       *
*  datal1:            pad   1  KPIO      (4467,2433)   *
*  datal2:            pad   2  KPIO      (4467,2767)   *
*  datal3:            pad   3  KPIO      (4467,3100)   *
*  datal4:            pad   4  KPIO      (4467,3433)   *
*  datal5:            pad   5  KPIO      (4467,3767)   *
*                                                       *
*  addr15:            pad   6  KPIO      (3800,4400)   *
*  addr14:            pad   7  KPIO      (3467,4400)   *
*  addr13:            pad   8  KPIO      (3133,4400)   *
*  addr12:            pad   9  KPIO      (2800,4400)   *
*  addr11:            pad  10  KPIO      (2467,4400)   *
*  addr10:            pad  11  KPIO      (2133,4400)   *
*  addr9:             pad  12  KPIO      (1800,4400)   *
```

```
*  addr8:              pad  13  KPIO       (1467,4400)   *
*  addr7:              pad  14  KPIO       (1133,4400)   *
*  gnd1:               pad  15  KPGND      (800, 4400)   *
*                                                        *
*  addr6:              pad  16  KPIO       (133, 3767)   *
*  addr5:              pad  17  KPIO       (133, 3433)   *
*  addr4:              pad  18  KPIO       (133, 3100)   *
*  unused1:            pad  19  KPIO       (133, 2767)   *
*  unused2:            pad  20  KPIO       (133, 2433)   *
*  unused3:            pad  21  KPIO       (133, 2100)   *
*  sysreset:           pad  22  KPIO       (133, 1767)   *
*  sysclk:             pad  23  KPIO       (133, 1433)   *
*  addr3:              pad  24  KPIO       (133, 1100)   *
*  addr2:              pad  25  KPIO       (133,  767)   *
*                                                        *
*  addr1:              pad  26  KPIO       (800,  133)   *
*  addr0:              pad  27  KPIO       (1133, 133)   *
*  read/write:         pad  28  KPIO       (1467, 133)   *
*  data0:              pad  29  KPIO       (1800, 133)   *
*  data1:              pad  30  KPIO       (2133, 133)   *
*  data2:              pad  31  KPIO       (2467, 133)   *
*  data3:              pad  32  KPIO       (2800, 133)   *
*  data4:              pad  33  KPIO       (3133, 133)   *
*  data5:              pad  34  KPIO       (3467, 133)   *
*  data6:              pad  35  KPIO       (3800, 133)   *
*                                                        *
*  data7:              pad  36  KPIO       (4467, 767)   *
*  data8:              pad  37  KPIO       (4467,1100)   *
*  data9:              pad  38  KPIO       (4467,1433)   *
*  data10:             pad  39  KPIO       (4467,1767)   *
*  vdd:                pad  40  KPVDD      (4467,2100)   *
*                                                        *
**********************************************************
```

The routing of buses to the pads was an important step of construction, because no software existed in either system of layout tools used for this design that could verify correct connectivity. Signal tracing of the final composite mask layout was the only method by which inter-connectivity could be verified.

A copy of the final composite mask layout for the TORO 680-16 is available from Dr. Andrzej Rys, Associate Professor of Electrical Engineering at Kansas State University, Manhattan, Kansas. This copy is only available in the California Intermediate Form (CIF), using layer definitions from MOSIS. The ASCII file is 820 kilobytes in length.

3.0 Design Simulation

As mentioned above, the design was simulated using the VIVID CAD tool FACTS, using circuit parameters extracted and given by MOSIS for the 3-micron bulk p-well SCMOS process technology. The I/O pad and the three-state standard cell were characterized using SPICE 3A7. The simulations described in this section were run to show the functionality of the large functional blocks, which did, in turn, verify the correctness of the block's inter-connectivity. These simulations were run under no-load conditions to "speed up" the simulation time on the SUN 3/60. Delays obtained from these simulations were only used as approximations to delays expected for the final design. Final system simulations, given in section 4.0, give the delays recorded by FACTS for the completed design.

Simulations for some standard cells were run to get an idea of how much of a capacitive load they could handle, and then capacitances were watched carefully as the design grew so that this maximum capacitance would not be exceeded. If this capacitance was exceeded, a non-inverting driver cell with twice the driving capability was used. It was understood early on that three-state buffers would have to be constructed to handle the large

capacitance of the passive main internal bus. It was determined from simulation using FACTS that the majority of standard discrete cells could handle loads of up to 1 pF before rise/fall times became excessive (larger than 20 ns). After the completion of the final design in the VIVID editor, all capacitances on long buses for data and control were checked to make sure that these maximum load values were not exceeded. Recall again that it was not the purpose of this project to fully characterized the standard cell library.

Because of the large number of points to be recorded during simulation, and the length of the simulations, data recorded was kept for each 10 ns plot step. This limit made it difficult to determine accurately the rise/fall times of the outputs for each of the large functional blocks; the VIVID simulation plot tool SIMPLOT often rounded to the nearest 10 ns step when calculating the propagation delay. However, at this stage it was only necessary to determine the functionality of each functional block. Some internal nodes for the functional blocks were watched, but again, only to verify the integrity of the design. Capacitances were closely watched. Average power was also recorded, but, as will be discussed later, was only used to get a feel for how large

a current density could be expected in the power buses. Recall that power consumed in a CMOS design is linearly proportional to the frequency of operation.

3.1 Register Set Tests

Four tests were performed on the register set. The first test was conducted to show the independence of the MAR, TMP, and IR registers, to show that those registers only loaded while enabled, and only on the rising edge of the clock. The second test was conducted to show the independence of each bit of the MAR, TMP, and IR. The third test was conducted to show that the A and X registers could be loaded independently of the MAR, IR,and TMP. The fourth test was conducted to show that the A and X registers were effectively isolated from the main internal bus by the three-state buffers used in the functional block. The register set passed all tests. Plots of the results of the above tests and a description for each simulation is given in Appendix B at the end of this report.

3.2 ALU Tests

Twelve tests were performed to show the functionality of the ALU. The tests were performed for the AND, OR, XOR, CMP, SHL/ROR, SHL/ROL, INC, DEC, COM, TST, and ADD

instructions. Also, a test was performed to check the three-stating of the ALU output to the main bus. For most of the tests, the inputs to the ALU remained the same. Control signals were changed to allow the ALU to perform its various functions. However, for the ADD, CMP, and DEC tests, inputs were created such that the results would give the maximum carry propagation delay for the ALU as well as show ALU functionality. For instance, for the DEC instruction, an input of 0 was given to show that an FFFF would result. The carry was also shown to give the appropriate 0 output.

The above result may seem incorrect; for a DEC operation that results in a carry, the appropriate carry output should be recorded as a 1. Indeed, the carry bit input multiplexing, located in the control functional block, inverts this output during the CMP, TST, and SUB instructions. Other such boundary conditions for the ALU were performed for the ADD instruction, for example, incrementing FFFF. During this instruction, the carry output recorded by the status register was not inverted.

For these worst case boundary conditions, the approximate ALU delay was noted. Recall that the outputs of the ALU were not loaded, and additional loading in the final design gave somewhat slower delays. This additional

delay was in the 10-15 ns range when compared with final design simulation data; the additional delay was due to capacitive loading at the ALU three-state outputs. From results given from FACTS for the DEC and ADD simulations during boundary-condition tests, the approximate delay from data inputs to the ALU three-state output was 100 ns. Historically, the propagation delay for the ALU is usually the longest delay for the system, and thus, is the limiting factor in the maximum frequency of operation. Again, the plots and command files for this set of simulations appear in Appendix B.

Note that the write register was not tested in this functional block. This register was added to the final design after final design simulation verified that write timing to external memory would not be possible without it. This register was simulated for functionality in the tests performed in Section 4.0.

3.3 Program Counter Tests

The testing of the program counter was relatively straight-forward. Tests of three types were performed. First, tests for each four-bit macro of the assembled 16-bit counter was performed to show that each macro had been properly inter-connected to the clock, reset, and ripple-

carry outputs. The second set of tests were run to check for proper loading of the counter from the load inputs, that is, the main internal bus. Data was loaded such that after loading, the counter would properly enable the next cascaded macro. This set of tests was also performed to show that the counter was capable of correctly counting from 0000 to FFFF. The third set of tests were performed to show the enable and disable characteristic of the counter as well as the ability to three-state the outputs of the counter from the main internal bus. Plots of these tests, along with descriptions for each test, appear in Appendix B.

3.4 Control Logic Tests

The simulations for the control logic were by far the most numerous. Several functions were performed by the control logic functional block. Four major types were performed. First, the TORO control signals were tested for each of the instruction classes: load, store, branch, and ALU, and each of the addressing modes: immediate, inherent, direct and indexed. Because the T phase clock and all other instruction register decode circuitry was included in this functional block, all that was necessary to test these control signals was to give the appropriate IR code to the control logic and run the system clock.

The second set of tests performed were for the ALU control signals. Again, this test was performed to check for the proper decoding of the instruction word. The loading of the status register was also simulated. For the negative and zero inputs, this check was trivial. The most complex part of this simulation was for the carry input multiplexing.

Another set of simulations was performed to check TORO branch control signal generation. However, the testing for the branch control signal was not exhaustive, given the large number of possible status register configurations and branch instructions. Plots of the results and one-page explanations for each simulation appear in Appendix B.

3.5 I/O Pad and Three-State Buffer Characterization

The majority of the system's timing depended on the propagation delays given for the I/O pads, and the three-state standard cell buffers connected to the main internal bus. For this reason, these two cells were fully characterized using SPICE 3A7 on the VAX 11/750.

As mentioned in Section 1.1, the required software for converting extracted circuit parameters from MAGIC standard cells into SPICE models was not installed on the

SUN computer used to conduct this design. The SPICE model used for the I/O pad was obtained from UC-Berkeley tools EXT2SIM and SIM2SPICE installed on a SUN system at the University Of New Mexico with the help of Dr. John Rasure[16]. MAGIC and the other UC-Berkeley tools are available from the NW Laboratory For Integrated Systems from the University of Washington. Worse-case transistor parameters from MOSIS were used. Circuit parameters for the three-state buffer were extracted by ABSTRACT, a tool from the VIVID Layout package from the Microelectronics Center Of North Carolina. Again, worst case transistor parameters from MOSIS were used.

These two circuits were tested for propagation delay while enabled, for high-Z-to-valid data, and for valid-data-to-high-Z for both low and high outputs. Figure 16 shows the output load used and the method by which delays were calculated. The load and measurement technique used was found in the 1978 National Semiconductor CMOS Data Book[17]. A summary of this technique appears in Figure 16. The SPICE decks used for these cells and ASCII plots of the relevant waveforms appear in Appendix A.

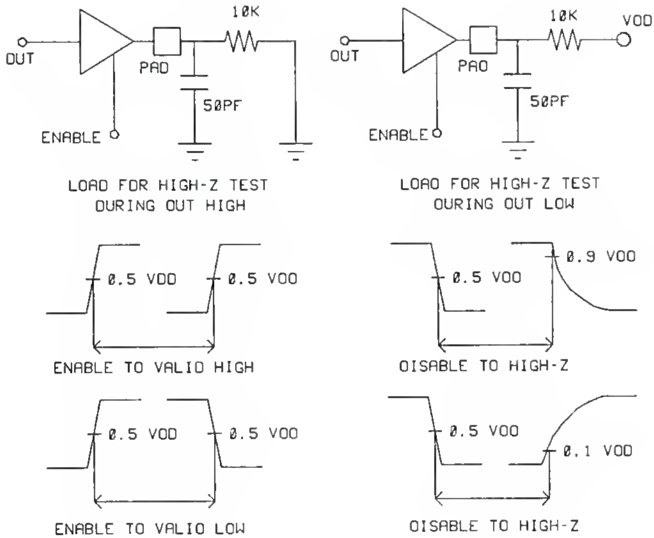

Figure 16: Three-State Buffer/Output Pad Measurement And Output Load Configurations

The following tables summarize the results found for the two cells. These results were used in calculating system timing requirements.

SIGNAL DESCRIPTION	CLK RISE/FALL 10 NS		CLK RISE/FALL 25 NS	
	DELAY	RISE/FALL	DELAY	RISE/FALL
IN-TO-OUT ENABLE HIGH CLOAD = 5 PF RLOAD = 10K	9.42	9.41	11.76	8.83
	11.76	8.23	14.71	8.83
ENABLE-TO-VALID HIGH	8.11		10.39	
DISABLE-TO-HIGH Z IN HIGH	10.81		11.88	
ENABLE-TO-VALID LOW	8.04		8.79	
DISABLE-TO-HIGH Z IN LOW	10.72		13.11	

TIMES GIVEN IN NANOSECONDS

Table 1: Three-State Buffer Standard Cell Data Sheet

SIGNAL DESCRIPTION		CLK SIGNAL RISE/FALL 10 NS				CLK SIGNAL RISE/FALL 25 NS			
		0.5 PF LOAD		5 PF LOAD		0.5 PF LOAD		5 PF LOAD	
		DELAY	RISE/FALL	DELAY	RISE/FALL	DELAY	RISE/FALL	DELAY	RISE/FALL
PAD-TO-IN ENABLE LOW	↓	3.3	–	5.1	2.5	5.9	–	6.4	4.6
	↑	2.5	–	4.2	4.2	2.5	–	3.7	5.5
PAD-TO-INBAR ENABLE LOW	↓	1.7	–	2.5	5.1	1.7	–	1.8	11.0
	↑	3.3	–	4.2	7.6	5.0	–	5.5	11.9
		50 PF LOAD				50 PF LOAD			
		DELAY	RISE/FALL			DELAY	RISE/FALL		
OUT-TO-PAD ENABLE HIGH	↓	25.5	5.8			30.8	6.2		
	↑	34.6	10.7			36.2	11.5		
ENABLE-TO-VALID HIGH		23.0				25.6			
DISABLE-TO-HIGH Z OUT HIGH		87.3				91.0			
ENABLE-TO-VALID LOW		13.2				15.7			
DISABLE-TO-HIGH Z OUT LOW		94.6				98.0			

Table 2: I/O Pad Simulation Data Sheet

The I/O pad has two inputs available, one inverted and one non-inverted. See Figure 15. The inverted input was used in this design to clock data into the write register. It also gave the best measure of what rise/fall times would be allowable for signals coming in from the pads to the design. The rise/fall times for the inverted input became excessive (> 15 ns) after moderate loading (5 pF) and excessive pad rise/fall times (40 ns). One can see from the above table for the I/O pad, however, that the non-inverted rise/fall times are acceptable for a 5 pF load with a pad rise/fall time of 25 ns. Output waveforms for this output load condition showed little degradation at the end of the transitions. For this reason, a maximum allowable input signal rise/fall time was set at 25 ns for all input signals. This result is mentioned again in Section 4.1.

4.0 System Verification And Characterization

In this section, the simulations used to verify the final design and the timing information calculated from those simulations are discussed. Simulations were done in four sets, one set for each of the instruction classes: load, store, branch, and ALU. These sets of data were used, along with propagation delay information for the I/O pads and three-state buffers, to calculate the timing requirements for the TORO 680-16. In order to get as accurate as possible the propagation delays through the TORO design, loads were added to the TORO outputs.

4.1 TORO Output Load Capacitances

First, the total load capacitances for the outputs from the TORO design were approximated. Recall that this was not done for the simulations performed on the four functional blocks that make the final TORO design. Loads were included here to simulate as closely as possible the design as it was constructed in the pad frame.

There were three buses involved in the final routing: one for the address output, one for data input/output, and one for control signal routing. The address bus was metal 1 and in the worse case has a length of about 1/3 the interior perimeter of the pad frame. It connected the MAR in the register set to output pads. For the data bus, the

routing was primarily metal 2 and in the worse case also had a length of about 1/3 the interior perimeter. This bus connected the main internal bus and write register to the pads. The control bus was a combination of metal 1, metal 2, and polysilicon. The control bus inter-connected the control signals generated in the control logic functional block to the control lines in the ALU, register set, and program counter. However, the VIVID tool ABSTRACT was able to extract capacitances for the control bus, given that it could be included in the guts of the final design. See Figure 1. Therefore, it was not necessary to calculate the load capacitances for that bus. Capacitances for the address and data buses could not be extracted, however, because those buses could not be included in the TORO guts. It was necessary to first approximate the lengths of those buses and calculate load by hand using processing parameters from MOSIS.

From the MOSIS service, the metal 1 layer capacitance was found to be $0.24(10)_{-4}$ pF/um^2, and metal 2 layer capacitance was $0.16(10)_{-4}$ pF/um^2. Recall that the pad frame was 6800 microns by 6900 microns. The pads used in the design were 640.5 micron in height. So, in calculating the interior perimeter of the pad frame, one must first subtract the heights of the pads from the length of each

side of the frame, then add those resulting length together. To find the area of one line in the bus, one must then divide this interior perimeter by three, and multiply the resulting length by the width of the wire, 4.5 microns. The resulting area must then be multiplied by the capacitance of the metal layer per unit area to obtain the load capacitance. Thus, for the address bus, the signal path layer capacitance was:

$$4.5 \times (\ 2 \times (\ 6800 - (\ 2 \times 640.5\ )\ \ ) + 2 \times (\ 6900 - (\ 2 \times 640.5\ )\ \ )\ ) \times 0.24(10)_{-4}\ /\ 3 = 0.802 \text{ pF}$$

Address Bus Capacitance: 0.802 pF

For the data bus, the calculation was similar, replacing $0.24(10)_{-4}$ with $0.16(10)_{-4}$. The resulting signal path layer capacitance was:

$$4.5 \times (\ 2 \times (\ 6800 - (\ 2 \times 640.5\ )\ \ ) + 2 \times (\ 6900 - (\ 2 \times 640.5\ )\ \ )\ ) \times 0.24(10)_{-4}\ /\ 3 = 0.534 \text{ pF}$$

Data Bus Capacitance: 0.534 pF

This signal path capacitance was added to each input and output in order to determine, first of all, whether or not maximum capacitances had been exceeded for cells in the TORO design. In addition, each of the input and output capacitances for the TORO680-16 were extracted using the

FACTS simulator, so that the capacitances could be used to determine the total capacitance for each TORO680-16 output node.

4.1.1 Output Load Capacitances

For the address outputs, the total load capacitance was computed by adding the above signal path capacitance to the output capacitance of the address outputs given by FACTS. For all address outputs, this capacitance was 0.261 pF. Also considered was the input capacitance of the OUT pin of the I/O pad. This capacitance was quite small, as given in the I/O pad SPICE deck in Appendix A. Thus the total load capacitance for the address outputs was 0.261 + 0.802 + 0.010 = 1.073 pF.

Address Output Load Capacitance: 1.073 pF

For the data outputs, the output capacitance given by FACTS was 0.121 pF for all outputs. This value added to the signal path load capacitance and the input capacitance of the I/O pad gave a total data output load capacitance of 0.121 + 0.534 + 0.010 = 0.665 pF

Data Output Load Capacitance: 0.665 pF

Three other outputs from the guts of the TORO680-16 were considered. Two are control signals for the enabling

and disabling of the address and data I/O pads, and the other was the read/write control signal. For this latter control signal, the output capacitance for the read/write from FACTS was given as 0.468 pF. The logic gate being used here is actually four non-inverting buffer cells in parallel. Thus, it is able to drive four times the load that a single buffer could, or about 4 pF. However, the read/write I/O pad is physically close to the output from the TORO, and for the purposes of computing the total load, this signal path capacitance was ignored. The total load capacitance for the read/write output was computed: 0.468 + 0.010 = 0.478 pF

Read/Write Control Output Load Capacitance: 0.478 pF

For this relatively small load, the driver was more than adequate. A large driver was used because, at the time of construction, it had not been determined where the read/write pad would be located in the pad frame.

For the address output enable control signal, 16 I/O pad enable control signals were driven. The input load capacitance for the I/O pad enable pin was computed to be 0.175 pF. This capacitance was computed by multiplying the total input gate area by the capacitance per unit area for the gate oxide as given from MOSIS. In addition, the metal

layer used to connect all these I/O pad enable inputs was metal 1. The length of this signal path is also much longer than for the address and data buses, about 2/3 the total interior perimeter. The output capacitance of the address output enable control signal was given by FACTS to be 0.870 pF. The total output capacitance of this signal was: (16 x 0.175) + (2 x 0.802) + 0.870 = 5.274 pF.

Address Output Enable Load Capacitance: 5.574 pF

The driver used here was also more that adequate for the load constructed. For the data output enable control signal, the resulting load capacitance was similar. Again, the signal path length was about 2/3 the interior perimeter and was routed on metal 1. From FACTS, the output capacitance for the data output enable control signal output was given as 0.831 pF. The total output capacitance for the data output enable: (16 x 0.175) + (2 x 0.802) + 0.831 = 4.834 pF.

Data Output Enable Load Capacitance: 4.834 pF

4.1.2 Input Load Capacitances

The input load capacitances for the TORO design was not required for the FACTS simulator. The FACTS simulator used clocks that were "almost ideal" drivers with 0 ns rise and fall times. It was still worth knowing system

clock, system reset, and data input capacitances, however, because these capacitances affect the rise and fall times for the output signal IN generated from the I/O pad. There were four input capacitances to consider: data input from pads, system clock, system clock bar, and system reset.

From the FACTS simulator, the data input capacitance for the TORO was given as 0.040 pF. This capacitance added to the data bus capacitance of 0.534 pF gave a total load capacitance for the IN output of the I/O pad of: 0.040 + 0.534 = 0.574 pF.

Data Input Load Capacitance: 0.534 pF

As shown in the data sheet for the I/O pad in Section 3.5, this capacitance was well within the operating range of the pad. The other inputs, the system clock and reset inputs, had much higher input capacitances. Given from FACTS, these capacitances were 4.299 pF and 1.123 pF, respectfully. Also, the I/O pad for these input signals were physically close, thus, the signal path layer capacitance was ignored. The load capacitance calculated for the pad inputs was the capacitances given by FACTS for the system reset and clock inputs.

System Clock Input Load Capacitance: 4.299 pF

System Reset Input Load Capacitance: 1.123 pF

The last input capacitance to consider was the system bar input which clocked data into the write register during write operations. This write register clock input was the only input connected to the INBAR output for the system clock pad. Thus, the input load for the system clock bar input was smaller than for the two cases above. The input capacitance given by FACTS for this input was 1.255 pF. Again, because the system clock bar input to the guts was physically close to the I/O pad, the routing external to the guts was ignored.

System Clock Bar Input Load Capacitance: 1.255 pF

It is important to note here that the output capacitance for the I/O pad was NOT included in the above calculations for the total TORO input load capacitances. This was because, as mentioned above, these capacitances were not used in the FACTS simulations. In FACTS, one must include all relevant capacitances, because any capacitance specified explicitly during a simulation for a node overrides the FACTS-calculated capacitance for that node. In comparison, the driving capability for the I/O pads was performed by SPICE, where additional capacitances may

simply be added to the SPICE deck. Table 3 gives a summary of the total load capacitances for the TORO 680-16 output nodes, and the load capacitances, as seen by the I/O pads, for the TORO input nodes. These capacitances as given in Table 3 were used in the simulations that follow.

LOAD CAPACITANCE DESCRIPTION	C_{LOAD}
ADDRESS OUTPUT	1.073 PF
DATA OUTPUT	0.665 PF
READ/WRITE OUTPUT	0.478 PF
ADDRESS ENABLE OUTPUT	5.574 PF
DATA ENABLE OUTPUT	4.834 PF
DATA INPUT	0.534 PF
SYSTEM CLOCK INPUT	4.299 PF
SYSTEM CLOCK INBAR	1.255 PF
SYSTEM RESET INPUT	1.123 PF

Table 3: Total Load Capacitances For TORO680-16 Final Assembled Design

4.2 System Timing

Simulations were performed on the guts of the TOR0680-16, using the above load capacitances, to check for final design functionality and performance. From these simulations, the propagation delays for address output from system clock were obtained, as well as the propagation delays for the address output enable control signal, the data output enable control signal, and the read/write control signal. Other information about internal propagation delays for the registers, control signals, and register-transfer were also obtained. This other information was used in determining the maximum obtainable operating frequency.

In determining the system timing, it was first necessary to consider the system timing waveforms for each instruction class and addressing mode. This is shown in Figures 17 through 23 below. The numbered delays shown in the figures of system timing correspond to values computed later in this section and are defined in Table 4.

Delay Number	Propagation Delay Definition
(1)	System Clock Frequency
(2)	Clock-To-Address-Valid
(3)	Clock-To-Address-Not-Valid
(4)	Data Set Up Time Before Clock
(5)	Data Hold Time After Clock
(6)	Clock-To-Read/Write-Low
(7)	Clock-To-Write-Data-Valid
(8)	Clock-To-Write-Data-Not-Valid
(9)	Clock-To-Read/Write-High
(10)	System Clock High Time

Table 4: Definitions Of Numbered Delays For TORO 680-16 System Timing

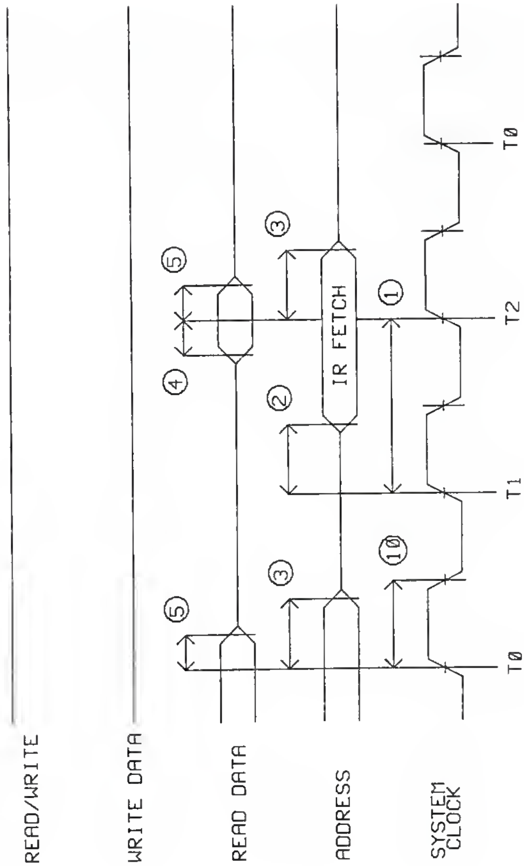

Figure 17: System Timing During Inherent Addressing For ALU Instructions

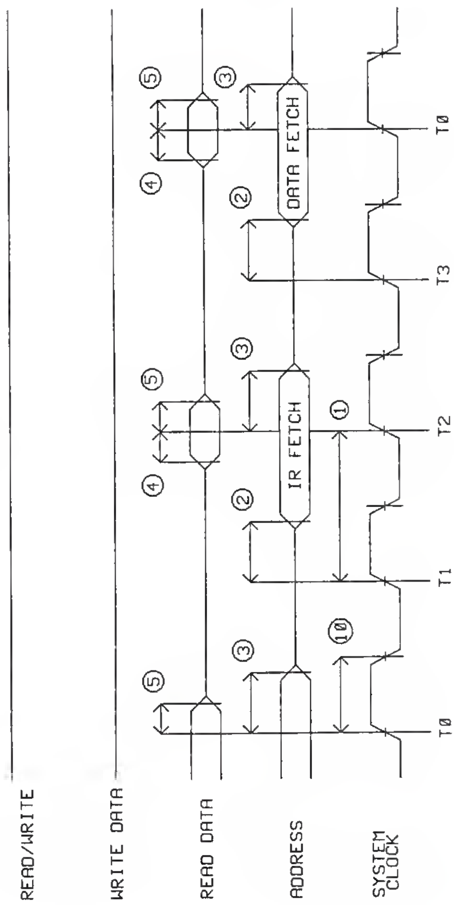

Figure 18: System Timing During Immediate Addressing For Load/Branch Instructions

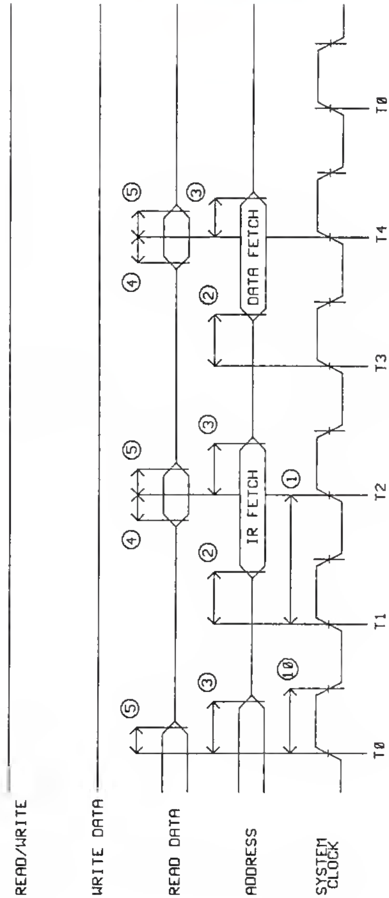

Figure 19: System Timing During Immediate Addressing For ALU Instructions

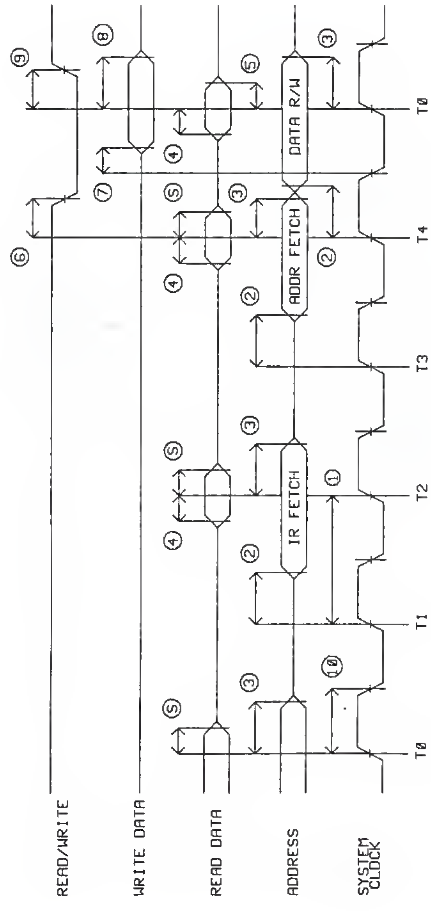

Figure 20: System Timing During Direct Addressing For Load/Store/Branch Instructions

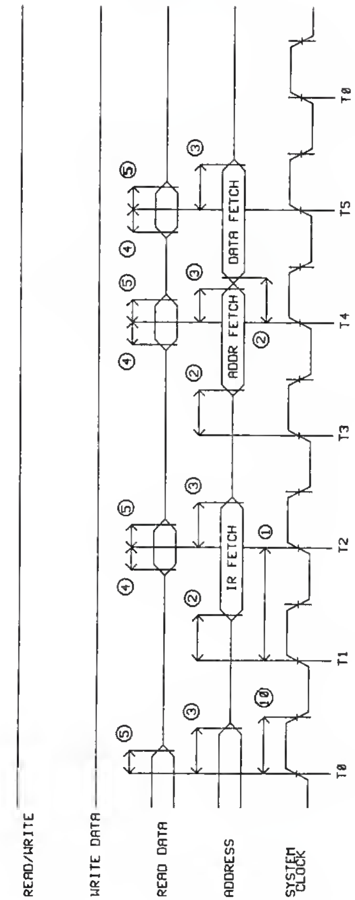

Figure 21: System Timing During Direct Addressing For ALU Instructions

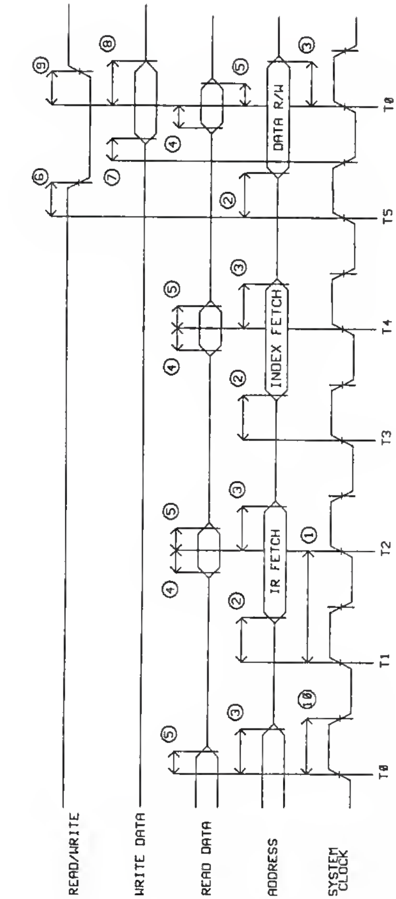

Figure 22: System Timing During Indexed Addressing For Load/Store/Branch Instruction

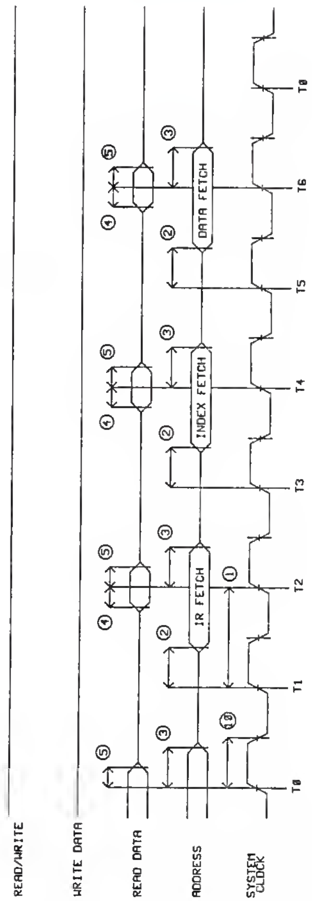

Figure 23: System Timing During Indexed Addressing For ALU Instructions

From the figures, one can see the cycles taken by the TORO to execute each instruction class in each addressing mode. For three of the addressing classes, the timing was straight-forward; two complete system clock cycles were used to accomplish register transfer and read/write from memory. The only "quirk" in the timing occurred during direct addressing where only one cycle was used. The direct addressing mode required that two conditions for clock-to-valid-address system timing be considered.

Once the waveform timing for the TORO machine was determined, simulations were performed. There were two simulations total, one for the load immediate instruction and one for the store indexed instruction. More simulations were planned, but time could not be made available. As a note, for the SUN 3/60 used for this thesis, the amount of CPU time these simulations consumed was 38 and 48 hours, respectfully. The FACTS simulator reported 11,496 transistors and 5270 nodes in the guts of the final design. Plots of the simulations, along with the simulation files used, are given in Appendix B.

Propagation delays calculated by the VIVID tool SIMPLOT for the above simulations are given in Appendix B. The propagation delay information was used in making the following timing specifications for the TORO. In

calculating the system timing, propagation delays were often "rounded up" to the nearest 5 ns for worst case, and "rounded down to the nearest 10 ns for best case, propagation delays. This was done to help insure that the timing requirements thus given would insure proper operation of the TORO machine. Recall, however, that all simulations were performed using worst case transistor parameters. In turn, some "best case" assumptions may not be conservative enough to insure proper operation. Additional simulations would have to be performed to verify the timing in those cases. Again, time could not be made available for these simulations. However, according to Pucknell and Eshraghain[18], one should expect that actual measured delays for the fabricated device will be from two to three time SLOWER than for delays obtained from simulation, even when using worse case parameters.

One assumption in system timing must be noted here. The set-up and hold times for the register set were not computed explicitly from simulation. Recall that the standard cell library for this design was not characterized. However, through simulation for the final design, minimum set-up and hold times were "discovered" during indexed addressing. The hold time for the MAR was determined by watching the propagation delays in the TORO

after the rising edge of the system clock. The delays in question were for signals that control the loading of the MAR. The rising edge allowed the MAR to latch onto the effective address computed by the ALU during indexed addressing.

From simulation, it was shown that the load control signal for the MAR, one of only three possible recipients of the ALU output data, went from a active high state to an inactive low state in 30 ns. The clock-to-output propagation delay for the register itself was 10 ns. Recall that the registers use feedback from their outputs to accomplish data loading. The three-state control signal for the ALU went inactive after the rising edge of the clock in 25 ns. The ALU three-state buffers went from valid-data-to-high-Z at best 10 ns later. Thus, data was valid at the inputs to the MAR for 35 ns, 5 ns longer than the load control signal. The simulation also showed that the data from the ALU had been successfully latched by the MAR. So for the rest of the system timing calculations, the minimum hold time for the registers was assumed to be the length of the data valid condition: 35 ns. The set-up time was assumed to be approximately 1.5 times this hold time, or 50 ns.

Register Minimum Set-Up Time: 20 ns

Register Minimum Hold Time: 40 ns

(1) System Clock Frequency

The system clock frequency was determined from the simulation performed for the Store Indexed instruction. During indexed addressing, the contents of the TMP were added to the contents in the X register to compute the indexed address. In the same cycle as this operation, the ALU output data, that is, the indexed address, was allowed to propagate through the ALU three-stating onto the main internal bus. This data propagation delay, from the loading of the TMP through the ALU and three-stating, was one of the longest propagation delay experienced by the system, when the addition created a sum of 0000 with overflow into the carry bit register. The carry bit from the ALU was stable 105 ns after the rising edge of the clock. The effective address was stable and available on the main internal bus in 110 ns. However, the longest delay from the ALU was for the zero bit. Recall that it was generated using 16 cascaded two-input OR gates. The zero bit delay was 135 ns.

Hold times for the register loading the status bit data must be added. From above, this hold was assumed to be 35 ns. Thus, the minimum clock cycle frequency is 1/170

ns, or 5.88 MHz. Rounding down to insure proper operation:

System clock frequency: 5.0 MHz.

(2) Clock-To-Address-Valid

As mentioned above, there were two cases to be considered in the clock-to-address-valid system timing. First, in the direct addressing mode, the delays considered were the sum of the clock-to-MAR-out (10 ns) and the OUT-to-pad propagation delays (40 ns). As can be seen from Figures 21 and 22, the address for the data to be fetched or written was immediately loaded into the memory address register. As had been determined from simulation, the address output enable control signal did not fall at any time in-between the two address outputs, that is, no three-stating occurred EXCEPT FOR THE ALU DIRECT INSTRUCTION. The consequences of this hazard will be discussed momentarily. Assuming first that no hazard occurred during the execution of the instruction, the clock-to-address-valid system timing was:

Clock-To-Address-Valid (min) = 50 ns.

For the ALU direct instruction, a hazard occurred and created a pulse 7.5 ns wide. The hazard recovered, that is, the address enable output returned to its high state

30 ns after the rising edge of the system clock. This hazard can be seen in Figure B32 in Appendix B. Because of this hazard, it was necessary to perform simulations using SPICE on the I/O pad cell to determine the effect the hazard might have on the propagation delay for enable-to-valid-address system timing.

Simulations showed that the hazard did have some effect on the computing of this delay. The hazard occurred before the pad output could change state. Because the pad three-state was disabled before the output of the pad could change state, and because the hazard on the the address output enable control signal was a short pulse, the output of the pad remained at its previous state. The pulse prevented the output from changing state because the pulse was short enough to prevent discharging of the capacitance of the bus, the condition necessary to give the high-impedance state. Thus, no hazards appeared on the I/O pad output because of the hazard present in the address output enable control signal, and no three-state condition existed because of the pulse. The only effect was additional propagation delay for the clock-to-valid-address system timing.

The simulation of the hazard condition showed that while the pulse was low, the I/O pad had time to set up for the enable-to-valid-data condition. That is, as soon as the hazard pulse went high, the only delay the pad experienced was in "turning off and on" the large driver transistors in its output stage. This delay was just a few nanoseconds less than the enable-to-valid-data delay given for the I/O pad in Table 2. Thus, the total delay for the clock-to-valid-address system timing during direct addressing was the sum of the clock-to-active-address-enable-control-signal delay after the hazard (30 ns) and the enable-to-valid-data for the pad (25 ns).

Clock-To-Address-Valid (hazard) = 55 ns.

For the second case, the address is loaded into the MAR while the I/O pad is in high-Z. The delay from the MAR is 10 ns, which is much less than the following delays, so it may be ignored. The address enable control signal from the rising edge of the clock was rounded up 35 ns in the worse case. The delay for enable-to-valid-data for the pad was rounded up to 30 ns. Thus, the maximum clock-to-valid address was their sum, or:

Clock-To-Address-Valid (max) = 65 ns.

Another point should be made here. Some fabricated devices will, by Murphy's Law, have an address enable control signal that will have a hazard that occurs AS THE OUTPUT OF THE PAD IS CHANGING STATE. This implies that for some finite time, the voltage on the pad will be at a metastable value. Only after the rising edge of the hazard will the output of the pad have the opportunity to continue to change from high-to-low or low-to-high. For most external peripheral and memory, this will not be a problem as long as the device is not enabled (or selected) before the pad output has been allowed to become stable. It may also be to a designer's advantage who uses the TORO to use an external address latch.

(3) Clock-To-Address-Not-Valid

As for the above, there were two cases to consider. One for the direct addressing mode, and one for all others. For the first case,the data at the outputs of the address register changed on the rising edge of the system clock. But, since the pad will not three-state for most instructions, the only delay to consider was the delay for the sum of the delays from the address register and the pad while the output was enable: the same as for the clock-to-address-valid (min) above:

Clock-To-Address-Not-Valid (min) = 50 ns.

When using the ALU direct instructions, however, a hazard in address output enable will occur and slow the system timing:

Clock-To-Address-Not-Valid (hazard) = 55 ns.

For all other addressing modes, the MAR outputs do not change on the rising edge of the clock. The two delays to sum here were the delay for the address enable control signal and the disable-to-high-Z for the pad. These delays were 35 ns and 100ns, respectfully. The clock-to-address-not-valid system timing was:

Clock-To-Address-Not-Valid (max) = 135 ns.

(4) Data Set-Up Time Before Clock

For this system timing, only the delays calculated for the I/O pad and the three-state buffer were required. The data pad IN and INBAR outputs to the guts of the design are isolated from the outside world and the OUT input of the I/O pad by a three-state buffer. This buffer was "opened" at the rise of the first clock and "closed" on the rise of the second. The read control signal controlled the enabling of this buffer and had a delay of 25 ns. The buffer was fully enabled 10 ns later. Thus, the main internal bus was "open" to the external data bus in

35 ns, 15 ns before the memory address was valid in the best case. It was guaranteed then that the main internal bus was available long before the rising edge of the next clock cycle.

The total delay for the data set-up time was the sum of the propagation delays for the pad-to-IN signal for the pad, the IN-to-OUT signal for the three-state buffer, and the set-up time for the register. From above, the set-up time was 50 ns. Rounded up to the nearest 5 ns, the pad-to-in delay was 10 ns and the IN-to-OUT delay was 15 ns. The set-up time for the data before the clock was:

Data Set-Up Time Before Clock: 75 ns.

(5) Data Hold Time After Clock

From above, the hold time for the registers was 35 ns. However, the read control signal delay was 25 ns and the three-state buffer disabled 10 ns later in the best case. Requiring that the data be stable for the duration of the sum of the read control signal and three-state buffer disable-to-high-Z delays, the register hold time would be met.

Data Hold Time After Clock: 35 ns

It is understood that this time could be made shorter, given the propagation delays for signals through

the I/O pads. However, it was desired to consider the very worst case, given the lack on concrete information on the hold time for the register set.

(6) Clock-To-Read/Write Low

From the timing diagrams in Figures 20 and 22, note that the write data control signal was clocked against the falling edge of the system clock. The following sequence of events occurs during the write cycle.

On the rising edge of the clock in the write cycle, the read/write and address output enable signals are generated. The propagation delay for the read/write and address output enable signal were both 30 ns. Recall that the pad for the read/write signal to external memory was permanently enabled. From Table 3, the OUT-to-pad delay was rounded up to 35 ns, thus, the total delay for the read/write signal was the sum of these two delays.

Clock-To-Read/Write Low: 65 ns

Note that this is the same delay that was given for the clock-to-address-valid system timing. From the read-write cycle timing for the memory referenced, the address set-up time before read/write was met. The memory referenced was the TMS 2114 NL[19]. The minimum set-up time allowed for

this memory was 0 ns.

(7) Clock-To-Write-Data Valid

Recall that the ALU functional block contained the write data register. On the rising edge of the clock for the write cycle, the data to be written to memory was placed on the main internal bus. On the falling edge of the clock for this cycle, the data was latched into the write data register, and the associated register delay was 10 ns. At the same time, the data output enable signal was generated with a delay of 30 ns. In addition, the enable-to-valid-data for the I/O pad was 30 ns. Thus, the data on the output of the write data register was present and stable before the data output enable signal. The sum of the data output enable signal and the enable-to-valid-data propagation delays for the I/O pad gave the clock-to-write-data-valid system timing.

Clock-To-Write-Data-Valid: 60 ns

For a design incorporating the TORO, the minimum system clock high time would be determined by subtracting this 60 ns from the maximum output disable time after write enable low for the memory used, and adding that result to the clock-to-read/write low timing given above, 65 ns.

(8) Clock-To-Write-Data-Not-Valid

The very next rising edge of the system clock after write data becomes valid controlled the removal of this data from the pads. The load signal for the write data register and the data output enable signal was removed. The data written was still present at the OUT inputs of the pads. Thus, the sum of the disable-to-high-Z delay for the pads and the data output enable control delay gave the total delay for the clock-to-write-data-not-valid system timing. These delays were 100 ns and 30 ns, respectfully.

Clock-To-Write-Data-Not-Valid: 130 ns

(9) Clock-To-Read/Write High

The sum of the read/write control signal propagation delay and the propagation delay for the I/O pad from the OUT input to the pad gave the required clock-to-read/write high system timing. The delays were both 35 and 30 ns, respectfully.

Clock-To-Read/Write High: 60 ns

Note that if the above timing was subtracted from the clock-to-write-data-not-valid, the hold time for the write data, in terms of the external memory, would be 70 ns. This hold time would, in turn, be more than adequate for most static memory. If the above timing was subtracted

from the clock-to-address-not valid system timing, the resulting address hold time would be 75 ns. Again, this hold time would be more than that required for most memory.

(10) System Clock High Time

As mentioned above, this timing requirement was mostly dependent on the memory used in a design incorporating the TORO. The minimum system clock high time would be determined by subtracting the 60 ns clock-to-write-data-valid from the maximum output disable-time-after-write- enable-low for the memory used, and adding that result to the clock-to-read/write low timing given above, or 65 ns. Stated more briefly, add 5 ns to the maximum output disable-time-after-write-enable-low for the memory used to get the required system clock high time. Internally, the minimum clock high time was only critical for the d-flip flops used. Because of the long high and low times required for external memory, these times for the d-flip flops would be met.

(11) Input Signal Rise/Fall Times

System clock and reset rise/fall times were determined from the simulations performed on the I/O pad. The limiting factor was the rise/fall times generated at

the I/O pad INBAR outputs to the guts, specifically the write data register. From these I/O pad simulations, it was determined that the longest rise/fall time for any signal propagating into the TORO from the pads should not be more than 25 ns. For signals with longer rise/fall times, the rise/fall times generated at the INBAR outputs of the I/O pad became degraded at the ends of the transition.

Input Signal Rise/Fall Time: 25 ns.

(12) System Reset Low Hold Time

During power-up and system reset, the control signals for the TORO were the last signals to settle for the design. Recall that the control signals were generated from the output of the phase clock T, the d-flip flops of which are asynchronously reset. The first operation of the TORO after reset was to load the MAR with the output of the program counter, 0000. Thus, the enable-to-valid-data propagation delay for the program counter three-state buffers (15 ns) and the set-up time for the memory address register (50 ns) were added to the longest control signal propagation delay after reset (50 ns) to determine the minimum low hold time for the system reset.

System Reset Low Hold Time: 115 ns

As long as the system reset signal was held low, the TORO would not run, as the d-flip flops in the phase clock T were asynchronously reset. This implied that some delay between the rising edge of the reset signal and the first rising edge of the system clock should be considered, to insure that the d-flip flops in the registers latched their input data correctly. However, no circuitry was added to insure that this minimum delay between the two signals would be met. This reset circuitry should added externally to synchronize the reset and clock signals.

The following table is the final system timing as computed from the above calculations. It can be referred to numerically from the figures for the system timing waveforms given in Figures 17 through 23.

	SYSTEM TIMING DESCRIPTION	MIN	TYP	MAX
1	SYSTEM CLOCK FREQUENCY	5.0 MHZ	—	—
2	CLOCK-TO-ADDRESS-VALID	50 NS	65 NS	—
3	CLOCK-TO-ADDRESS-NOT-VALID	50 NS	135 NS	—
4	DATA SET-UP TIME BEFORE CLOCK	75 NS	—	—
5	DATA HOLD TIME AFTER CLOCK	35 NS	—	—
6	CLOCK-TO-READ/WRITE-LOW	—	65 NS	—
7	CLOCK-TO-WRITE-DATA-VALID	—	60 NS	—
8	CLOCK-TO-WRITE-DATA-NOT-VALID	—	130 NS	—
9	CLOCK-TO-READ/WRITE-HIGH	—	60 NS	—
10	SYSTEM CLOCK HIGH TIME	SEE TEXT		
11	INPUT SIGNAL RISE/FALL TIME	25 NS	—	—
12	SYSTEM RESET LOW HOLD TIME	115 NS	—	—

Table 5: Table Of System Timing For The TORO680-16

4.3 Power Consumption And Maximum Power Rating

The VIVID simulator FACTS computed the average power and current for a design under simulation. The computation was averaged over the time of the simulation. Thus, the more transitions in a given amount of time, the more power that was consumed. Recall that the power consumed by a CMOS design is linearly proportional to the speed of operation. Power consumption was recorded for each simulation performed on the four functional blocks as well as for the final design.

The power calculated by FACTS was recorded for each of the four major functional blocks: the register set, the ALU, the control logic, and the program counter. The values recorded for each simulation appear in Appendix B in the simulation command files for that simulation. These values were used only as a relative measure of the current density in the power buses, to determine whether or not the buses were wide enough to prevent electromigration. However, if one reviews the speed of operation for the majority of the simulations, it can be noted that this speed was much higher than what can be expected for each functional block in the final design. Thus, the only set of power calculations that most accurately predicted the total power consumption for the guts of the TORO were the

values kept after simulations for the final design as a whole. These power values also appear in Appendix B for each simulation performed. The average power calculated:

$$P_{Internal} = 13.2 \text{ milliwatts}$$

Power consumption for the I/O pads was not calculated. At this time, SPICE 3A7 does not compute the power consumed by a circuit explicitly from simulation. In addition, this power consumption by the pads is dependent on the loads used. The maximum power consumed for the device is a function of the pad frame power and the internal power. The maximum amount of power that may dissipated, however, depends on the package used and the ambient operating temperature. Thus, this value is difficult to obtain from simulation. MOSIS can make available the thermal resistances for their packages and this maximum power could be predicted. However, the best measure for the power consumed would be after design fabrication.

5.0 Summary

In this report, the design methodology used in constructing the composite mask layout for the TORO680-16 16-bit microprocessor was discussed. The report began with an introduction on the origins of the TORO machine and

followed with a summary of the software and hardware used to facilitate the completion of the layout. The design was constructed and simulated for the 3 micron bulk p-well SCMOS process.

For this effort, two standard cell libraries were constructed, and thus, two designs were supported. One design was used primarily for simulation to verify the design's functionality and performance. The other design was the final composite mask layout. The construction began by defining to the logic level the essential MSI and LSI circuits for the TORO design. The size of the internal registers was increased from 8 to 16 bits, a write data register was added, and additional control circuitry was incorporated for read/write and pad three-state control. Macros of the resulting circuits were then constructed in one editor to check circuit functionality, then constructed in another editor for the final composite mask layout. Four major functional blocks were constructed: the register set, ALU, control logic, and the program counter. Once constructed and checked by simulation, the four blocks were floorplanned into a pad frame and connected to pads. The final design, excluding pads, was simulated, functionality and performance was again checked, and system timing for the machine was calculated. Power

consumption for the internal design was also calculated. Power dissipation was discussed, and it was determined that the best method by which this parameter could be obtained was by fabricating the device and measuring this power directly.

In addition, the integrity of the CAD tools used for this effort are currently under investigation. A "tiny chip" containing an eight-bit counter has been sent to a fabrication house that supports the 3 micron bulk p-well SCMOS process. Upon its return the chip will be fully characterized, and a qualitative measure will then be made of the standard cell library and tools used to construct this TORO design.

5.1 Design Construction Improvements And Performance Constraints

In considering the construction of the final design, some improvements and additions could be made. These improvements would make the design faster, more reliable in terms of system timing, and more compatible with microprocessor and peripherals already on the market.

First, some synchronization between the system reset and system clock signals would insure that registers upon reset would correctly load the required data. It was mentioned above that no circuitry was added to the TORO to

prevent the rising edge of the reset signal from coming too close to the rising edge of the system clock. Adding this circuitry would improve the reliability of the design. Some network of flip-flops and gates could be included in the control logic to perform this task. The control logic is the only place where real estate is available for this addition.

Also, the control signals as they are now constructed could be improved. Specifically, the write timing for the read/write signal to external memory is almost too fast, according to current simulation. As one may note, no data was given for simulations for best case conditions. Signals could be much faster that predicted, giving rise to undesirable conditions, especially for write timing. But, according to Pucknell and Eshraghain[20], these signals would probably be slower than predicted. Some delays added to the write control signals would give the design some flexibility in terms of processing parameters used, or allowed, during fabrication.

Possibly the largest improvement that could be made in speeding up the design on the whole would be in the redesigning of the ALU. Recall that the ALU was constructed from discrete gates. There are much better

ways to design arithmetic logic units in VLSI designs, ones that would decrease dramatically the size the ALU, and thus, the speed of data propagation. Some investigation was done in using these styles of ALU design, but the "silicon complied" method used in the construction of the ALU seemed a more direct and quicker way of designing an ALU that would performed the required operations. The cost of this "brute force" method of ALU design was a slower microprocessor.

In terms of the architecture of the TORO machine, a redesign that would use the ALU as the method by which the program counter is incremented would save a tremendous amount of real estate. It is understood that the purpose of using a synchronous sequential counter in this design was to make the design easier to teach in a introductory course in computer engineering. However, from a VLSI point of view, the size of the program counter is unwieldy when floorplanning the final design.

In addition, some circuitry could be added, given the unused pads in the pad frame, to include address and data clocks to the design. Currently, there are no signals generated by the TORO to inform a designer that addresses and data is valid, as for the E and Q clocks for the Motorola 6809[21]. In a design where the TORO is used with

existing peripherals, this circuitry would have to be constructed externally. This lack of informational control signals is the weakest feature of the TORO design.

6.0 References

[1] J. Devore, D. Hardin, A Computer Design For Introducing Hardware And Software Concepts, IEEE Transactions on Education, Vol. E-30, No. 4, November 1987, pgs. 219-226.

[2] Northwest Laboratory For Integrated Systems, VLSI Design Tools Reference Manual, Release 3.1, Department Of Computer Science, University Of Washington, Seattle, Washington, February 15, 1987. Chapters 1,2.

[3] Microelectronics Center Of North Carolina, Vertically Integrated VLSI Design Tools Reference Manual, Version 1.3, Research Triangle Park, North Carolina, 1987. Volumes 1,2,3.

[4] Information Science Institute, MOSIS User's Manual, Release 3.0, University Of Southern California, Marina del Rey, California, 1988. Chapters 1-12.

[5] D. Heinbuch, ed., CMOS3 Cell Library, Addison-Wesley Publishing Co., Reading, Mass., 1988, pg. 124-130.

[6] Microelectronics Center Of North Carolina, Vertically Integrated VLSI Design Tools Reference Manual, Version 1.3, Research Triangle Park, North Carolina, 1987. Volume 3.

[7] D.A. Pucknell, K. Eshraghian, Basic VLSI Design, Prentice-Hall of Australia Pty. Ltd., 1985. Sydney, Australia, pg. 267-280.

[8] N. Weste, K. Eshraghian, Principals Of CMOS Design, Addison-Wesley Publishing Co., Reading, Mass., 1985. pgs. 189-193.

[9] J. Devore, D. Hardin, A Computer Design For Introducing Hardware And Software Concepts, IEEE Transactions on Education, Vol. E-30, No. 4, November 1987, pg. 219.

[10] J. Devore, D. Hardin, A Computer Design For Introducing Hardware And Software Concepts, IEEE Transactions on Education, Vol. E-30, No. 4, November 1987, pg. 220.

[11] J. Devore, D. Hardin, A Computer Design For Introducing Hardware And Software Concepts, IEEE Transactions on Education, Vol. E-30, No. 4, November 1987, pg. 220.

[12] G.G. Langdon, Jr. Computer Design, Computeach Press Inc., San Jose, CA. 1982 pgs. 309-310.

[13] N. Weste, K. Eshraghian, Principals Of CMOS Design, Addison-Wesley Publishing Co., Reading, Mass., 1985. pgs. 320-322.

[14] The TTL Data Book, Texas Instruments, Dallas, TX., 1986. pg. 3-605.

[15] J. Devore, D. Hardin, A Computer Design For Introducing Hardware And Software Concepts, IEEE Transactions on Education, Vol. E-30, No. 4, November 1987, pg. 224.

[16] J. Rasure, Personal Communication, Department Of Electrical & Computer Engineering, University Of New Mexico, Albuquerque, New Mexico. February 28 - March 3 , 1989.

[17] CMOS Databook, National Semiconductor, Santa Clara CA., 1978. pg. 1-177.

[18] D.A. Pucknell, K. Eshraghian, Basic VLSI Design, Prentice-Hall of Australia Pty. Ltd., 1985. Sydney, Australia, pg. 207.

[19] _MOS Memory Data Book_, Texas Instruments, Houston, Texas, 1982. pgs. 87-90.

[20] D.A. Pucknell, K. Eshraghian, _Basic VLSI Design_, Prentice-Hall of Australia Pty. Ltd., 1985. Sydney, Australia, pg. 207.

[21] _8-Bit Microprocessor & Peripheral Data Book_, Motorola, Inc. Austin, Texas, 1983. pgs. 3-233 - 3-265.

[22] N. Weste, K. Eshraghian, _Principals Of CMOS Design_, Addison-Wesley Publishing Co., Reading, Mass., 1985. pgs. 189-193.

[23] N. Weste, K. Eshraghian, _Principals Of CMOS Design_, Addison-Wesley Publishing Co., Reading, Mass., 1985. pgs. 320-322.

[24] Microelectronics Center Of North Carolina, _Vertically Integrated VLSI Design Tools Reference Manual_, Version 1.3, Research Triangle Park, North Carolina, 1987. Volume 3.

7.0 Appendix A - Standard Cell Library

This appendix gives the mask composite layouts and transistor schematics for all the cells used in the TORO 680-16 microprocessor.

In this library, there are four groups of cells. All the groups share, however, several common attributes.

1) The overall height of the standard cells (excluding the pads and pad frame corners) is 70 lambda (105 microns). The Vdd and Vss buses are 12 lambda (18 microns) wide.

2) All inputs and output are available at the bottom and top of the cells. They are vertically routed on polysilicon, and interconnect the gates of the MOS complementary transistor pairs.

3) All p-transistors are 16 lambda (24 microns) wide and are oriented at the top of the cells, connected appropriately to Vdd. The n-transitors are 8 lambda (12 microns wide) and are oriented at the bottom of the cells, again connected appropriately to Vss. Transistors within the cell were interconnected via polysilicon and metal. This orientation of transistors was used so that the p-wells containing the n-transistors would be continuous for all abutted cells.

The four groups of cells contain the following standard cells and are described on the following pages: Discrete Standard Logic, Multiplexers, Flip-Flops, and the Pad Frame.

7.1 Discrete Standard Logic Cells

Below is a list of the discrete standard logic. These cells were the building blocks for the flip-flops cells, and comprise the majority of the TORO layout.

inv	**niv**	**nan2**	**nan3**	**nan4**	**exor**
invh	**nivh**	**nor2**	**nor3**	**nor4**	**tsdr**
		and2	**and3**	**and4**	
		or2	**or3**	**or4**	

The inv and invh are inverter cells. The h at the end of the invh cell name indicates that this cell is a buffer cell. It was designed to drive more that the regular inv cell. This was done by putting output transistors in parallel. This h convention is the same for the niv and nivh cells. They are non-inverting buffer cells.

The exor is an exclusive-OR gate. Note from the schematic and layout for the exor cell that it has four inputs. This cell is a two-input ex-OR gate, but because of a lack of real estate, additional routing must be added when the cell is used. Thus, the inputs that are labeled

with identical names must be inter-connected to accomplish the ex-OR operation.

The tsdr is a three-state driver cell. It features large output drivers so that large capacitive loads may be driven. In addition to the schematic, a SPICE deck using worst case transistor parameters from MOSIS is included. This deck was obtained from the execution of software from the VIVID CAD package, available from the Microelectronics Center Of North Carolina. Waveforms from simulation using the above mentioned SPICE deck are also included. This simulation was used to find the propagation delay for the cell from the IN pin to the OUT pin, three-stating enabled.

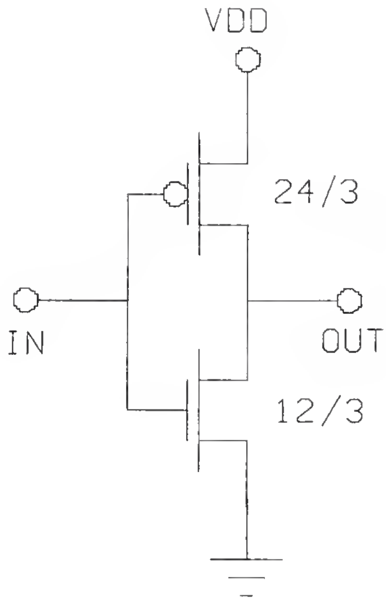

Figure A1: Transistor Schematic For The inv Cell

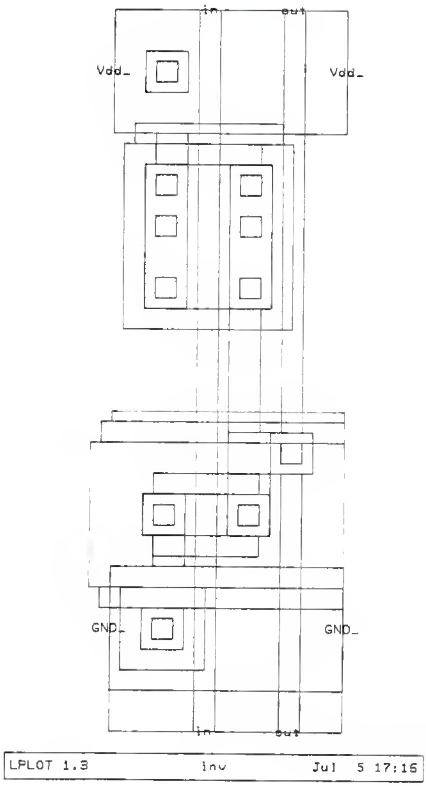

Figure A2: Composite Mask Layout For The inv Cell

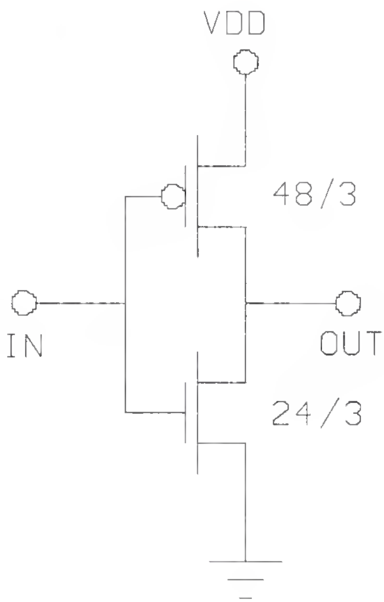

Figure A3: Transistor Schematic For The invh Cell

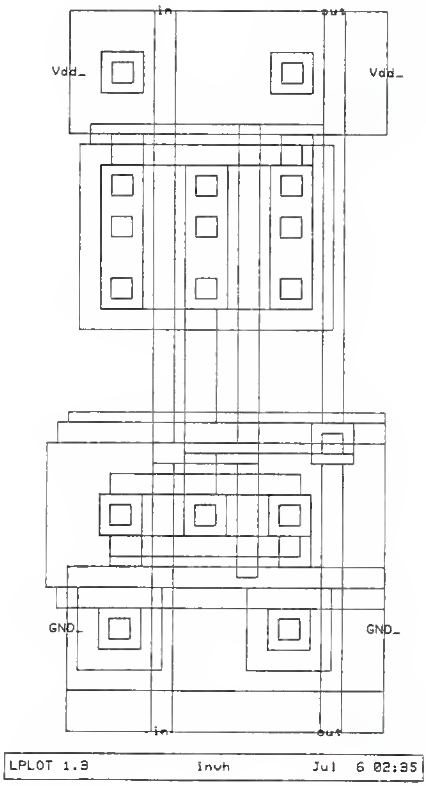

Figure A4: Composite Mask Layout For The invh Cell

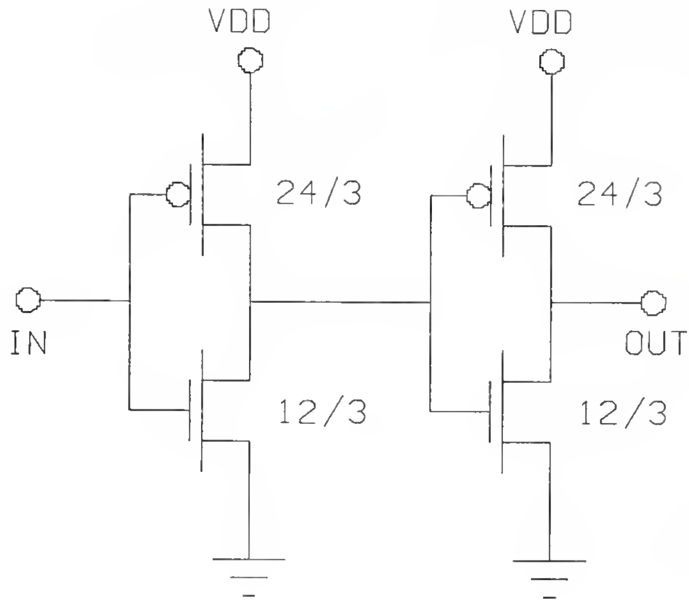

Figure A5: Transistor Schematic For The niv Cell

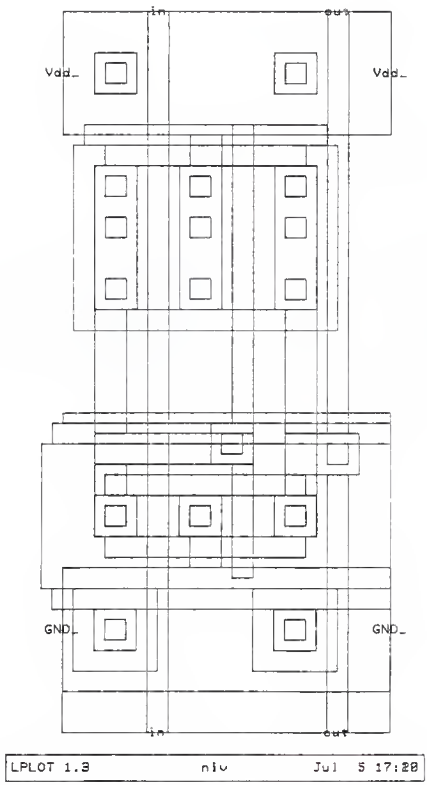

Figure A6: Composite Mask Layout For The **niv** Cell

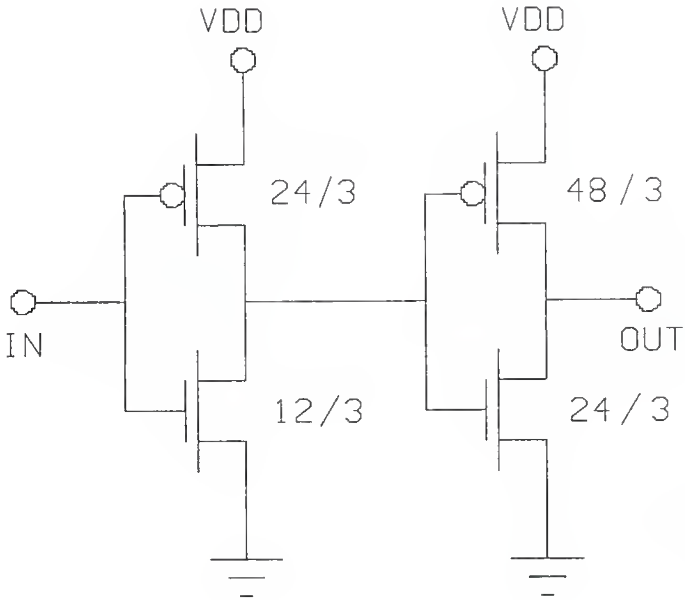

Figure A7: Transistor Schematic For The nivh Cell

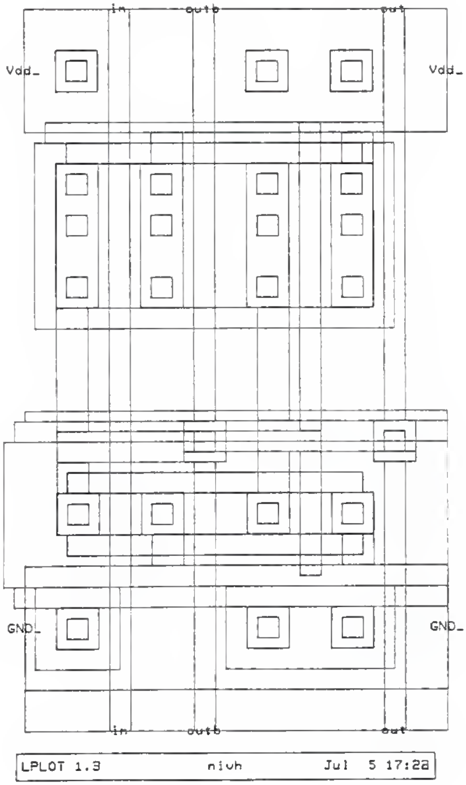

Figure A8: Composite Mask Layout For The **nivh** Cell

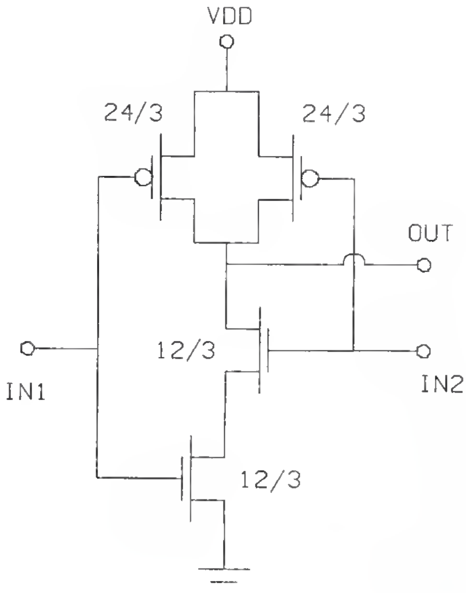

Figure A9: Transistor Schematic For The nan2 Cell

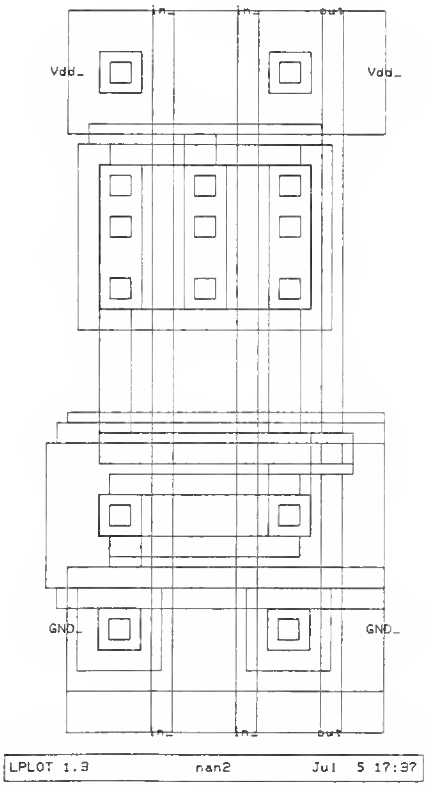

Figure A10: Composite Mask Layout For The nan2 Cell

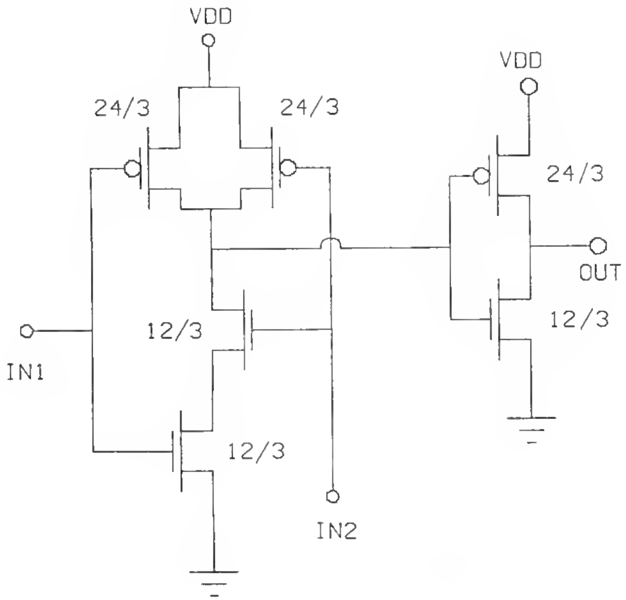

Figure A11: Transistor Schematic For The and2 Cell

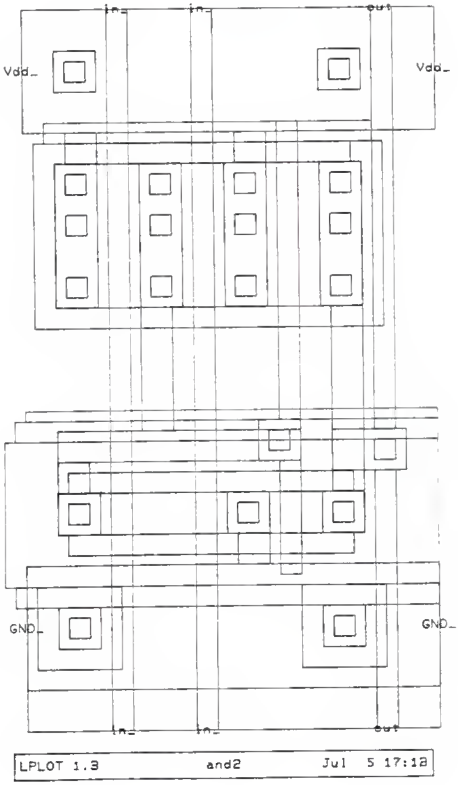

Figure A12: Composite Mask Layout For The and2 Cell

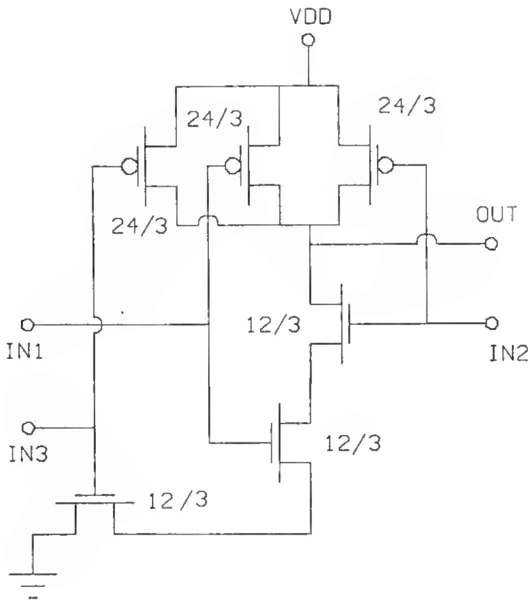

Figure A13: Transistor Schematic For The nan3 Cell

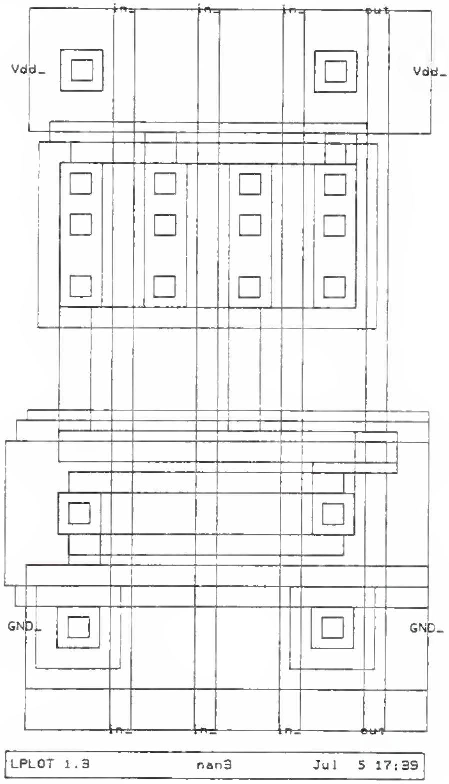

Figure A14: Composite Mask Layout For The nan3 Cell

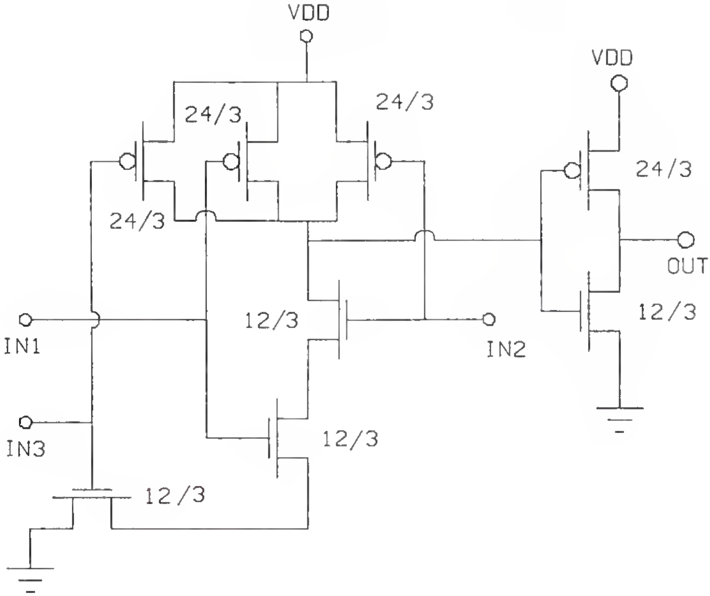

Figure A15: Transistor Schematic For The and3 Cell

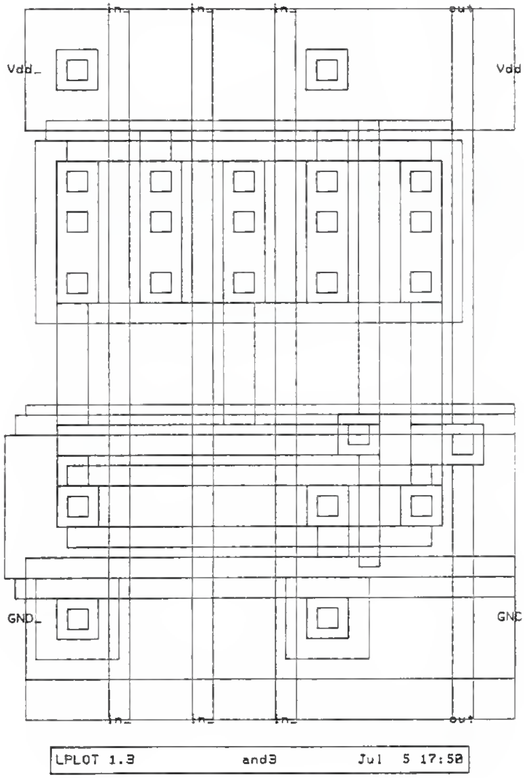

Figure A16: Composite Mask Layout For The and3 Cell

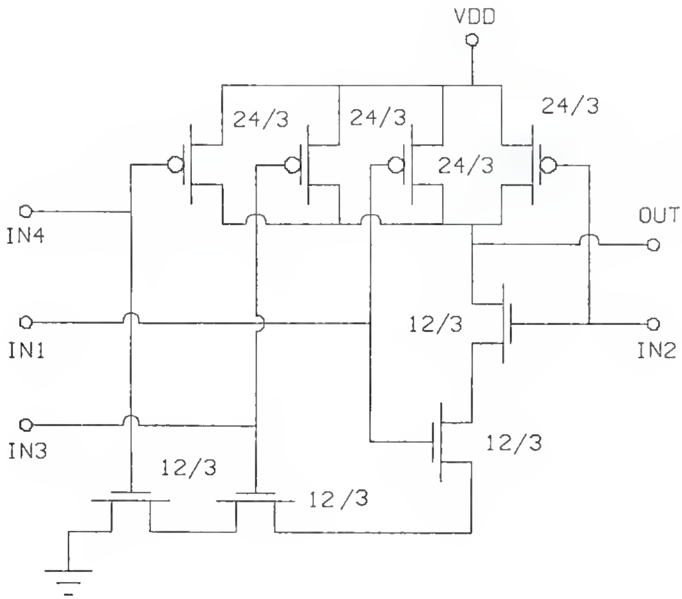

Figure A17: Transistor Schematic For The nan4 Cell

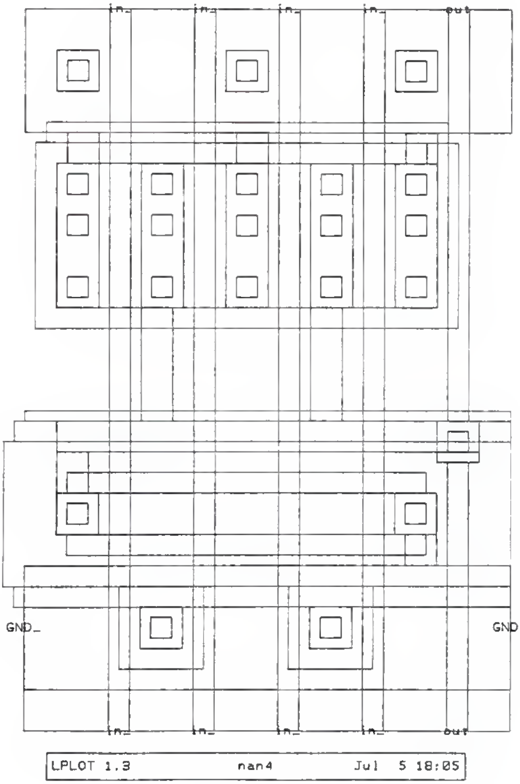

Figure A18: Composite Mask Layout For The nan4 Cell

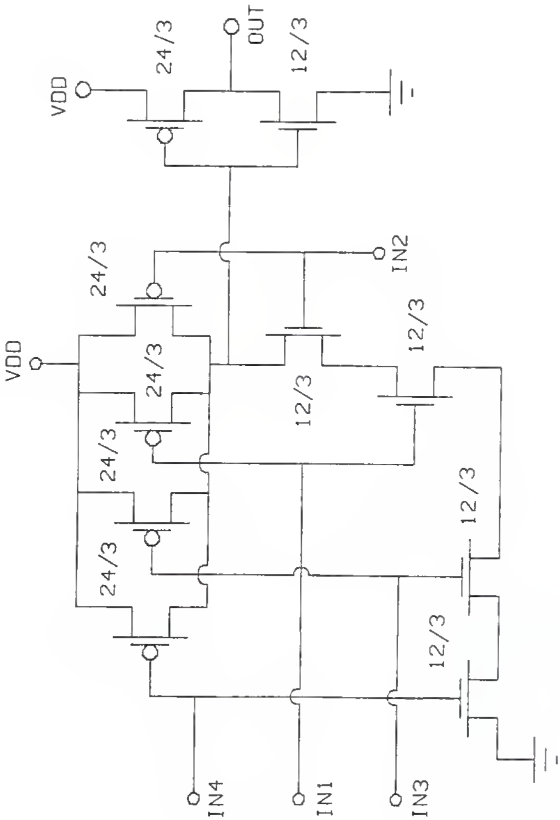

Figure A19: Transistor Schematic For The and4 Cell

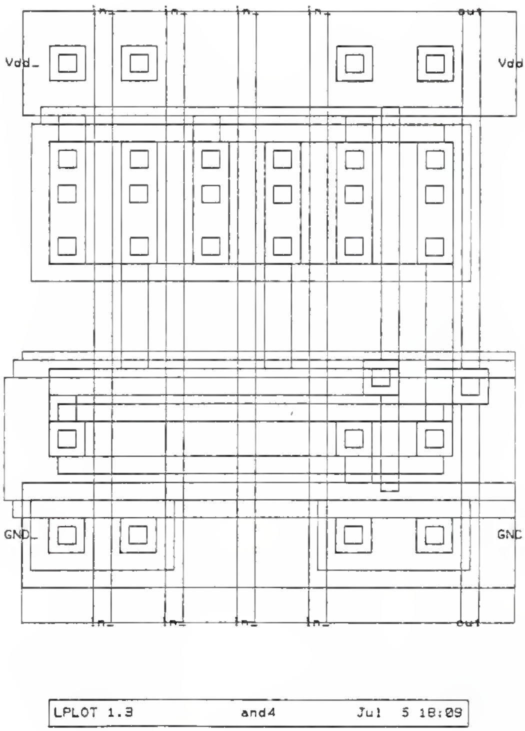

Figure A20: Composite Mask Layout For The and4 Cell

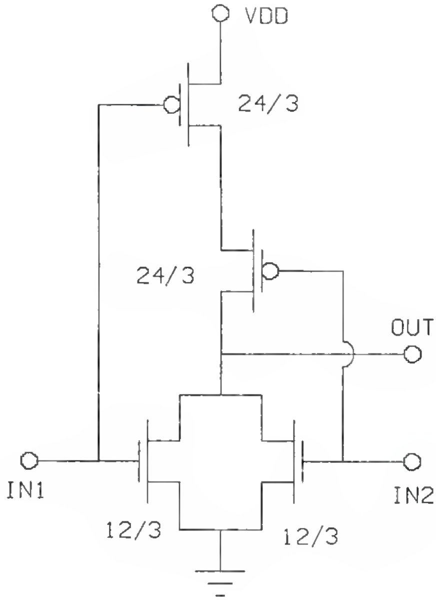

Figure A21: Transistor Schematic For The **nor2** Cell

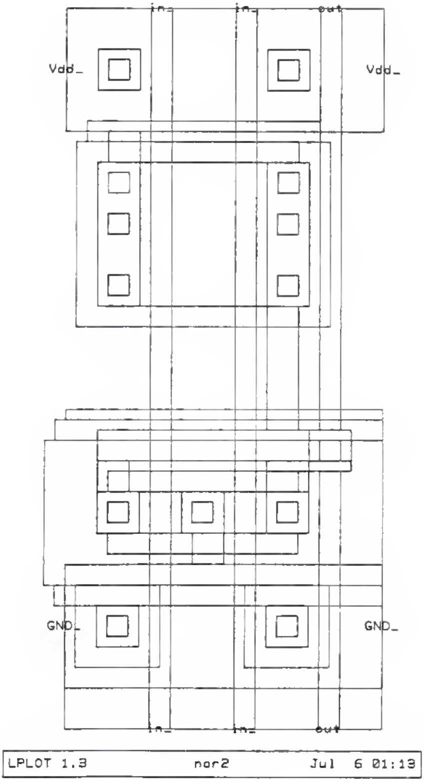

Figure A22: Composite Mask Layout For The nor2 Cell

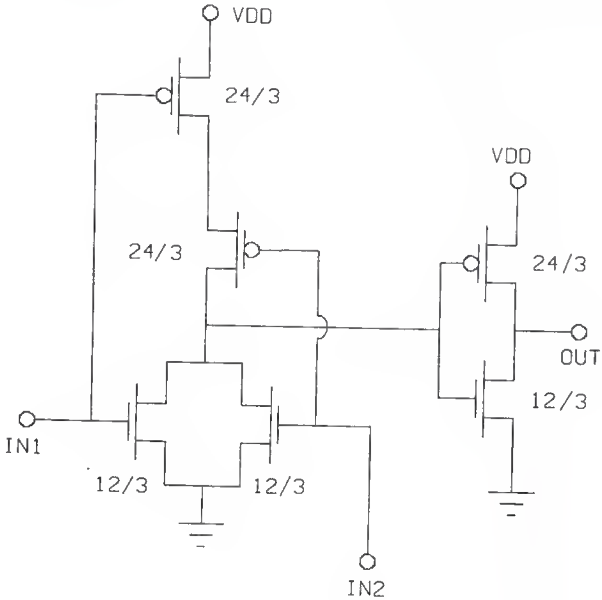

Figure A23: Transistor Schematic For The or2 Cell

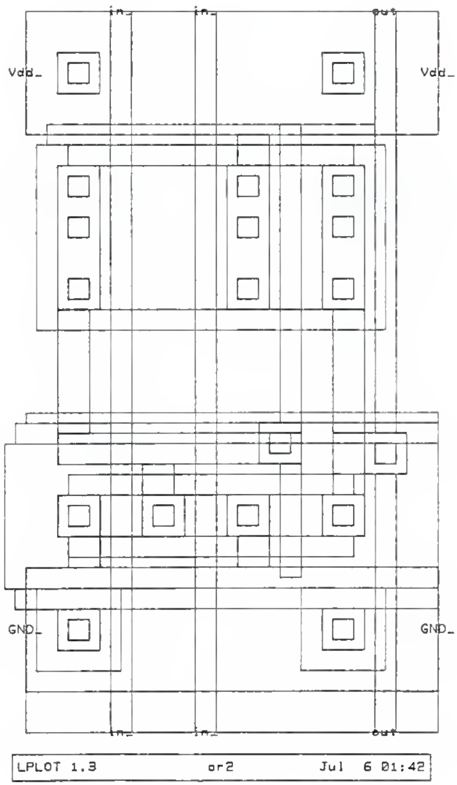

Figure A24: Composite Mask Layout For The or2 Cell

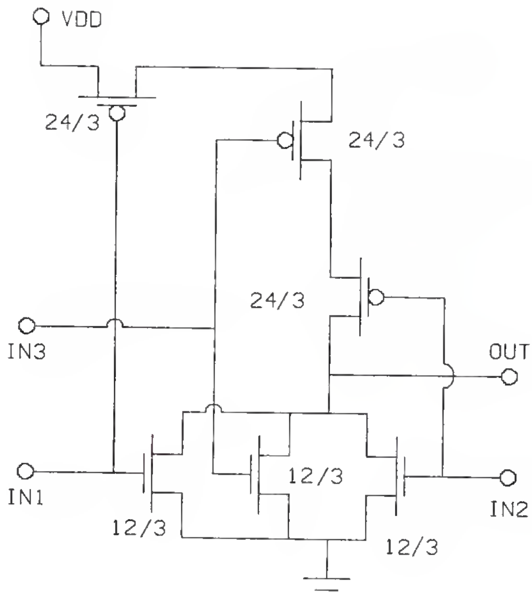

Figure A25: Transistor Schematic For The nor3 Cell

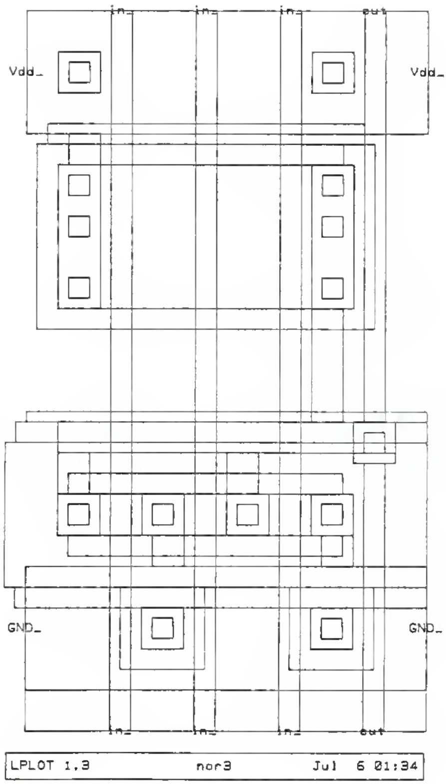

Figure A26: Composite Mask Layout For The nor3 Cell

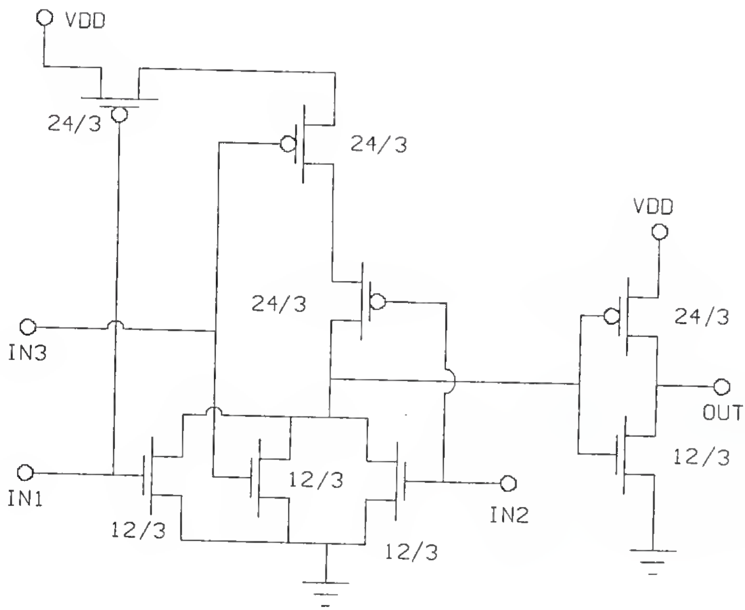

Figure A27: Transistor Schematic For The or3 Cell

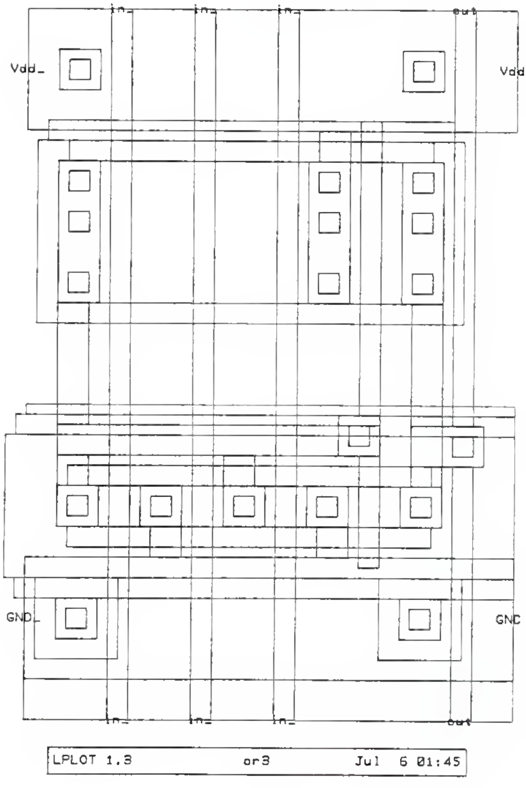

Figure A28: Composite Mask Layout For The or3 Cell

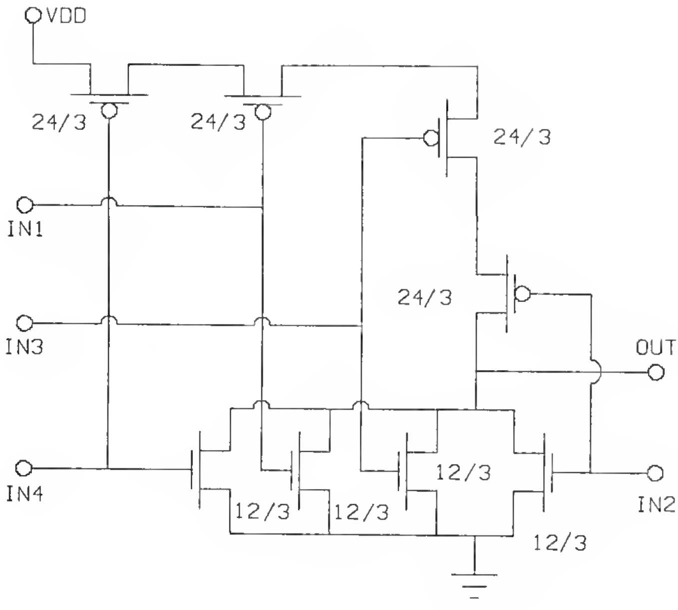

Figure A29: Transistor Schematic For The nor4 Cell

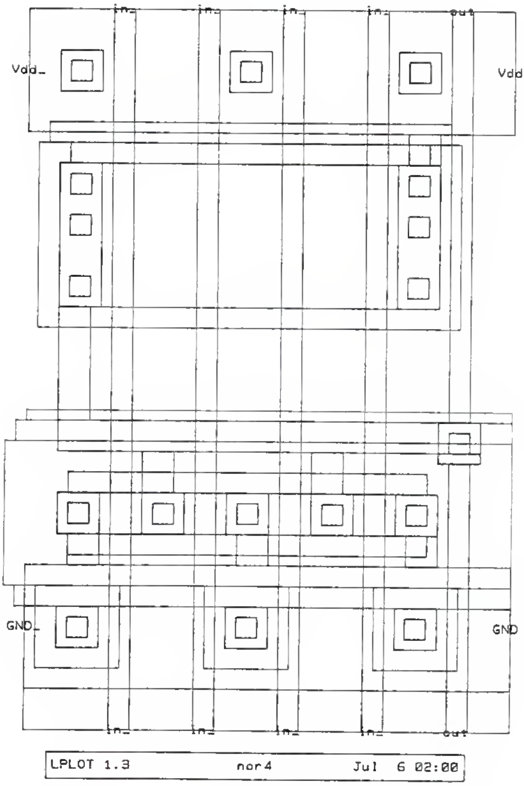

Figure A30: Composite Mask Layout For The nor4 Cell

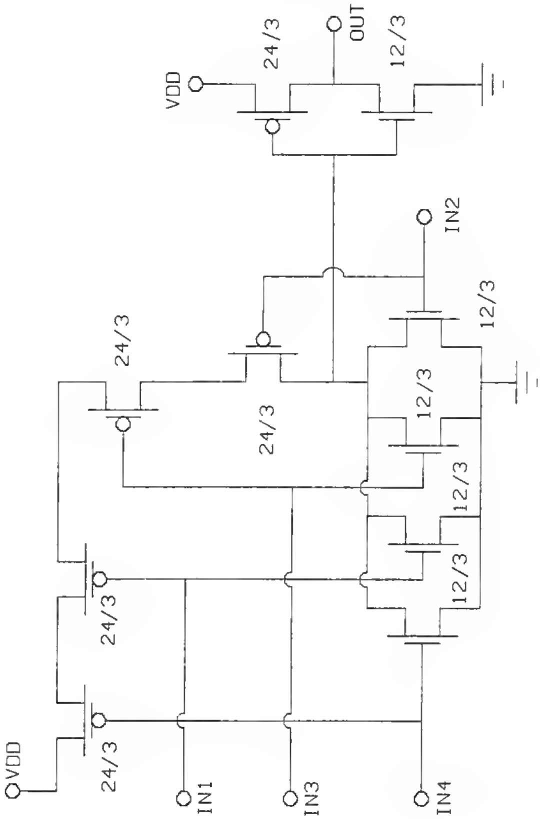

Figure A31: Transistor Schematic For The or4 Cell

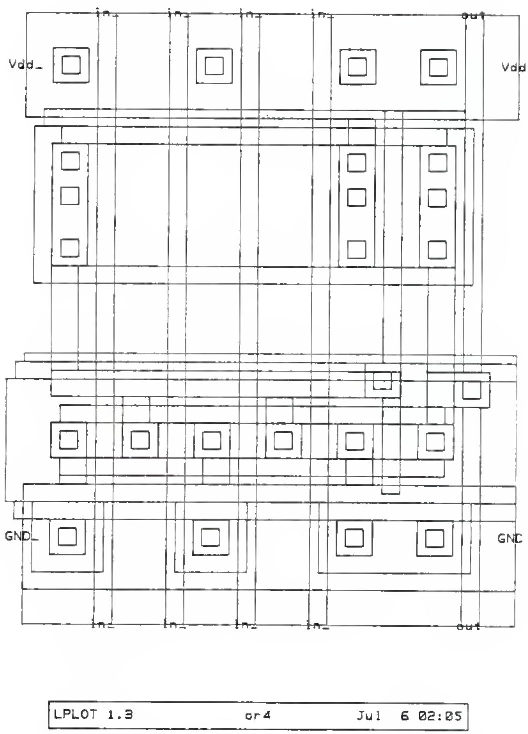

Figure A32: Composite Mask Layout For The or4 Cell

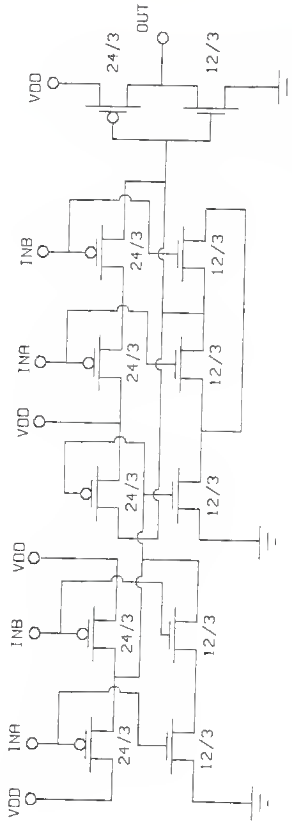

Figure A33: Transistor Schematic For The exor Cell

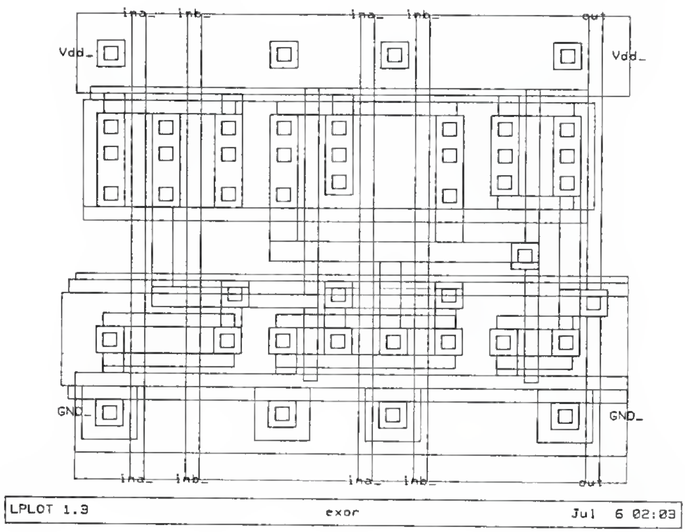

Figure A34: Composite Mask Layout For The exor Cell

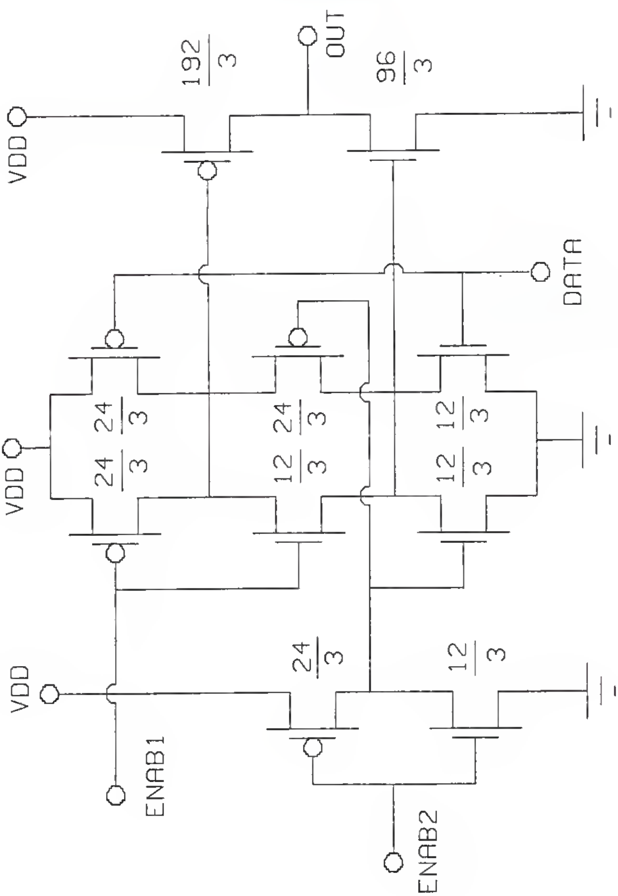

Figure A35: Transistor Schematic For The tsdr Cell

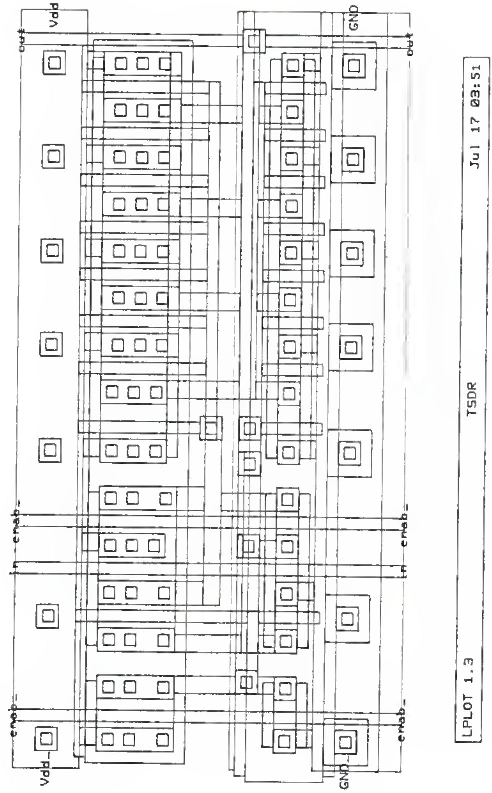

Figure A36: Composite Mask Layout For The tsdr Cell

```
TSDR
***************************************************************
*                                                             *
*   SPICE deck for the tsdr cell extracted from               *
*   the VIVID cell library using ABSTRACT                     *
*                                                             *
*                                                             *
*   This test is for enable1 and enable2 high,                *
*   propagation delay from IN to OUT.                         *
*                                                             *
* *************************************************************
*
*       Clock commands
*
Vss 1 0 0
Vdd 2 0 5
*
* THESE ARE WORST CASE VALUES FOR THE MOSIS 3u CMOS
* PROCESS
*
*
+ level=2.00              ld=0.320u           tox=550.e-10
+ nsub=1.0e+16            vto=1.000           kp=3.77e-05
+ gamma=1.50              phi=0.60            uo=600.0
+ uexp=1.000e-03          ucrit=999000.       delta=1.20
+ vmax=1.00e+05           xj=.600u            lambda=1.6e-02
+ nfs=1.200e+12           neff=1.00e-02       nss=0.0e+00
+ tpg=1.0                 rsh=30              cgso=5.2e-10
+ cgdo=5.2e-10            cj=2.2e-4           mj=0.5
+ cjsw=3.0e-10            mjsw=0.33
*
+ level=2.00              ld=0.480u           tox=550.e-10
+ nsub=1.120e+14          vto=-1.00           kp=1.260e-05
+ gamma=0.700             phi=0.60            uo=200.0
+ uexp=0.150              ucrit=1.6e+04       delta=1.9
+ vmax=1.0e+05            xj=.400u            lambda=4.7e-02
+ nfs=8.800e+11           neff=1.00e-02       nss=0.0e+00
+ tpg=-1.0                rsh=70              cgso=4e-10
+ cgdo=4e-10              cj=3.5e-4           mj=0.5
+ cjsw=2.0e-10            mjsw=0.33
*
mXd7 2 101 29 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
mXd4 29 101 2 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
mXd6 1 102 29 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
mXd5 29 102 1 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
```

```
m1 1 100 102 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m2 102 31 1 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m3 101 25 102 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m4 101 100 102 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m5 2 31 101 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m6 101 25 2 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
mXd3 2 101 29 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m7 100 23 2 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m8 1 102 29 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m9 100 23 1 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
mXd1 29 101 2 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
mXd2 29 102 1 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m10 2 101 29 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m11 1 102 29 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m12 29 101 2 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m13 29 102 1 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
mXd8 2 101 29 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
m14 29 101 2 2 penh l=3.0u w=24.0u
+ ad=144.0p pd=60.0u as=144.0p ps=60.0u
mXd9 1 102 29 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
m15 29 102 1 1 nenh l=3.0u w=12.0u
+ ad=72.0p pd=36.0u as=72.0p ps=36.0u
cenab2 23 1 7.98f
cenab1 25 1 7.98f
cout 29 1 45.78f
cin 31 1 5.70f
cI0 100 1 11.10f
cI1 101 1 40.20f
cI2 102 1 39.12f
*
```

```
*
*  OUT is node 29, IN is node 31, ENABLE1 is 25
*  ENABLE2 is node 23
*
*  Simulation Cards
*
*
Venabl 25 0 5V
Venab2 23 0 5V
Vin 31 0 PULSE(0,5,10ns,10ns,10ns,30ns,150ns)
*
*
*  Added capacitance for loading, and
*  resistance for tri-stating.
*
*
Cload 29 0 5PF
Rload 29 2 10K
*
*
*  Transient Card
*
*
 .tran 1ns 150ns
 .end TSDR
```

```
------------------------------------------------------------------
                                TSDR
Legend:  + = v(29)                       * = v(31)
------------------------------------------------------------------
TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
-----------|----|----|----|----|----|----|----|
 0.000e+00 .    X    .    .    .    .    .    .
 1.176e-09 .   +*    .    .    .    .    .    .
 2.353e-09 .   +*    .    .    .    .    .    .
 3.529e-09 .    X    .    .    .    .    .    .
 4.706e-09 .    X    .    .    .    .    .    .
 5.882e-09 .    X    .    .    .    .    .    .
 7.059e-09 .    X    .    .    .    .    .    .
 8.235e-09 .    X    .    .    .    .    .    .
 9.412e-09 .    X    .    .    .    .    .    .
 1.059e-08 .    +*   .    .    .    .    .    .
 1.176e-08 .    +   *.    .    .    .    .    .
 1.294e-08 .   +.    . *  .    .    .    .    .
 1.412e-08 .   +.    .    *    .    .    .    .
 1.529e-08 .   +.    .    .  * .    .    .    .
 1.647e-08 .   +.    .    .    .*   .    .    .
 1.765e-08 .   +.    .    .    .   *.    .    .
 1.882e-08 .    +    .    .    .    . *  .    .
 2.000e-08 .    +    .    .    .    .    *    .
 2.118e-08 .    . +  .    .    .    .    *    .
 2.235e-08 .    .    +    .    .    .    *    .
 2.353e-08 .    .    .   +.    .    .    *    .
 2.471e-08 .    .    .    .  + .    .    *    .
 2.588e-08 .    .    .    .    . +  .    *    .
 2.706e-08 .    .    .    .    .    +    *    .
 2.824e-08 .    .    .    .    .    . +  *    .
 2.941e-08 .    .    .    .    .    .  + *    .
 3.059e-08 .    .    .    .    .    .   +*    .
 3.176e-08 .    .    .    .    .    .   +*    .
 3.294e-08 .    .    .    .    .    .   +*    .
 3.412e-08 .    .    .    .    .    .   +*    .
 3.529e-08 .    .    .    .    .    .   +*    .
 3.647e-08 .    .    .    .    .    .   +*    .
 3.765e-08 .    .    .    .    .    .   +*    .
 3.882e-08 .    .    .    .    .    .   +*    .
 4.000e-08 .    .    .    .    .    .   +*    .
 4.118e-08 .    .    .    .    .    .   +*    .
 4.235e-08 .    .    .    .    .    .   +*    .
 4.353e-08 .    .    .    .    .    .   +*    .
 4.471e-08 .    .    .    .    .    .   +*    .
 4.588e-08 .    .    .    .    .    .   +*    .
-----------|----|----|----|----|----|----|----|
TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
```

```
TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
-----------|----|----|----|----|----|----|----|
 4.706e-08 .    .    .    .    .    .   +*    .
 4.824e-08 .    .    .    .    .    .   +*    .
 4.941e-08 .    .    .    .    .    .   +*    .
 5.059e-08 .    .    .    .    .    .   +*    .
 5.176e-08 .    .    .    .    .    .   +*    .
 5.294e-08 .    .    .    .    .    .   +*    .
 5.412e-08 .    .    .    .    .    .   +*    .
 5.529e-08 .    .    .    .    .    .   +*    .
 5.647e-08 .    .    .    .    .    .   +*    .
 5.765e-08 .    .    .    .    .    .   +*    .
 5.882e-08 .    .    .    .    .    .   +*    .
 6.000e-08 .    .    .    .    .    .   +*    .
 6.118e-08 .    .    .    .    .    .   +*    .
 6.235e-08 .    .    .    .    .    .   +*    .
 6.353e-08 .    .    .    .    .    .   +*    .
 6.471e-08 .    .    .    .    .    .   +*    .
 6.588e-08 .    .    .    .    .    .   +*    .
 6.706e-08 .    .    .    .    .    .   +*    .
 6.824e-08 .    .    .    .    .    .   +*    .
 6.941e-08 .    .    .    .    .    .   +*    .
 7.059e-08 .    .    .    .    .    .  *+.    .
 7.176e-08 .    .    .    .    .    *   +.    .
 7.294e-08 .    .    .    .    . *  .    +    .
 7.412e-08 .    .    .    .   *.    .    +    .
 7.529e-08 .    .    .    .*   .    .    +    .
 7.647e-08 .    .    .  * .    .    .    +    .
 7.765e-08 .    .    *    .    .    .    +    .
 7.882e-08 .    .  * .    .    .    .    +    .
 8.000e-08 .    *    .    .    .    .   +.    .
 8.118e-08 .    *    .    .    .    .  + .    .
 8.235e-08 .    *    .    .    .    . +  .    .
 8.353e-08 .    *    .    .    .   +.    .    .
 8.471e-08 .    *    .    .    .
 8.588e-08 .    *    .    .+   .    .    .    .
 8.706e-08 .    *    .+   .    .    .    .    .
 8.824e-08 .    *  + .    .    .    .    .    .
 8.941e-08 .    *+   .    .    .    .    .    .
 9.059e-08 .    X    .    .    .    .    .    .
 9.176e-08 .    X    .    .    .    .    .    .
 9.294e-08 .    X    .    .    .    .    .    .
 9.412e-08 .    X    .    .    .    .    .    .
 9.529e-08 .    X    .    .    .    .    .    .
 9.647e-08 .    X    .    .    .    .    .    .
-----------|----|----|----|----|----|----|----|
TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
```

```
TIME      -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
-----------|----|----|----|----|----|----|----|
 9.765e-08 .    X    .    .    .    .    .    .
 9.882e-08 .    X    .    .    .    .    .    .
 1.000e-07 .    X    .    .    .    .    .    .
-----------|----|----|----|----|----|----|----|
TIME      -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
```

7.2 Multiplexers

Two multiplexers were designed for the TORO 680-16 microprocessor: a 4x1 multiplexer and a 2x1 multiplexer. Both multiplexers are always 'on', that is, they do not have enable inputs.

mx2tl   mx4tl

Both designs were taken from Weste[22]. These cells are slow, but have inverter output stages to prevent output signal degredation for large load capacitances. Note also from the transistor schematic and layout that several inputs are identically labeled. As for the exor gate, these inputs must be inter-connected outside the cell for proper operation. These inputs were left unconnected in the multiplexer cells to reduce the control input capacitances. Of course, the cell has a large number of inputs as a consequence.

Four other cells fit under the catagory of multiplexers, namely, the carry look ahead cells used in designing the adder portion of the ALU.

lkad1   lkad2   lkad3   lkad4

These cells were also taken from Weste[23]. Transistor schematics for these cells were included in this section.

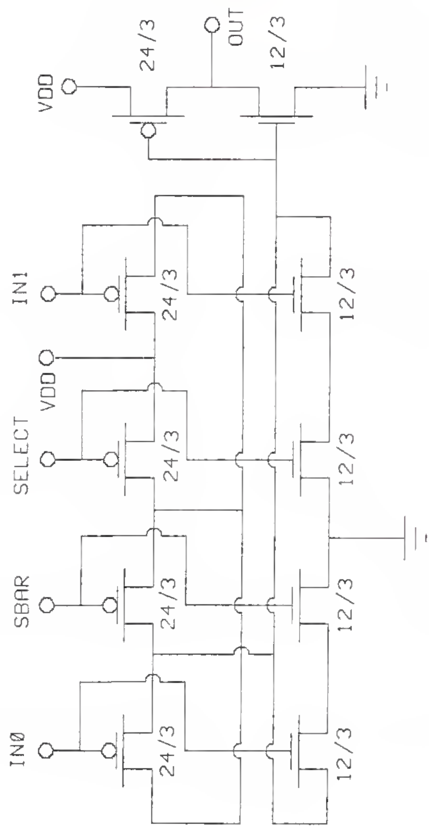

Figure A37: Transistor Schematic For The mx2tl Cell

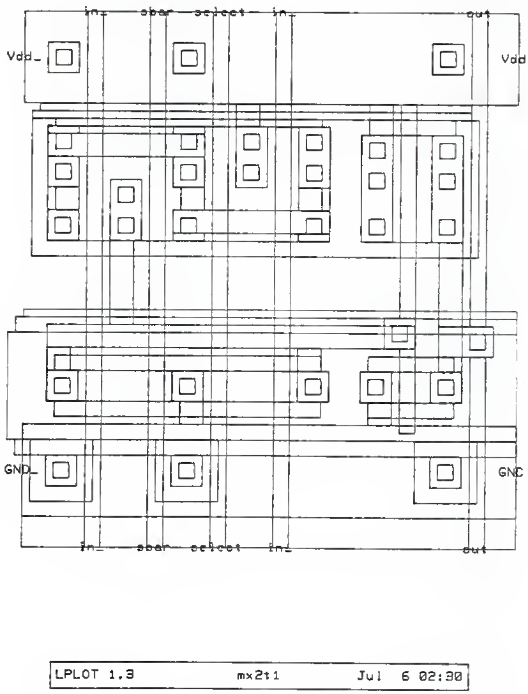

Figure A38: Composite Mask Layout For mx2tl Cell

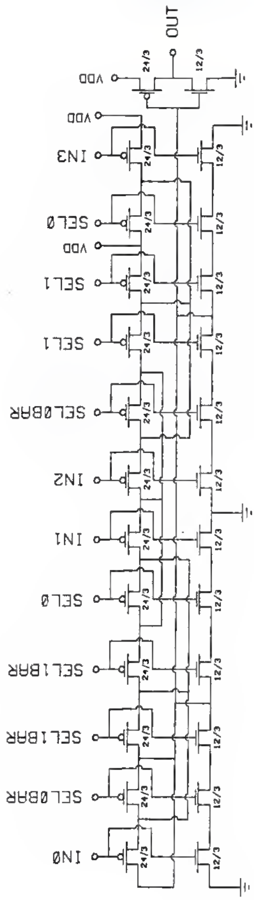

Figure A39: Transistor Schematic For The mx4tl Cell

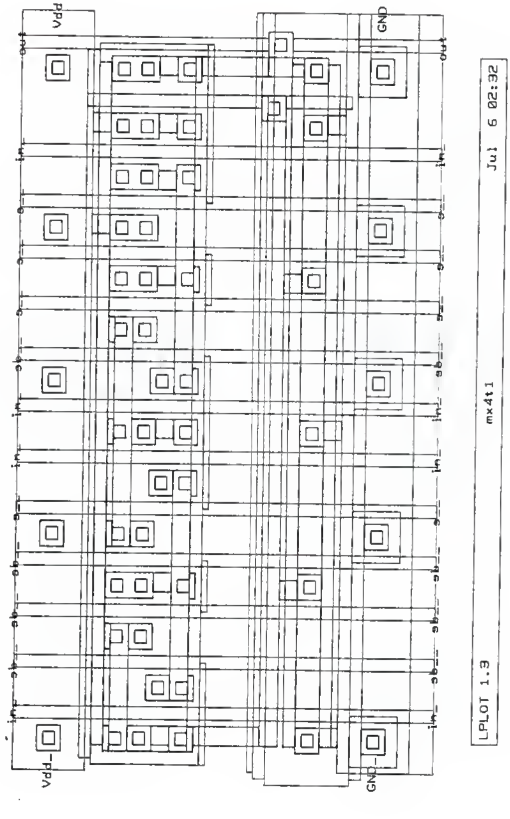

Figure A40: Composite Mask Layout For mx4tl Cell

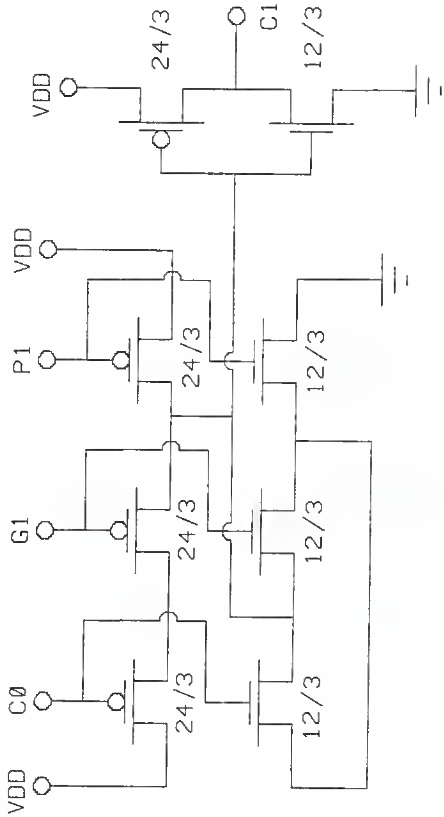

Figure A41: Transistor Schematic For The lkadl Cell

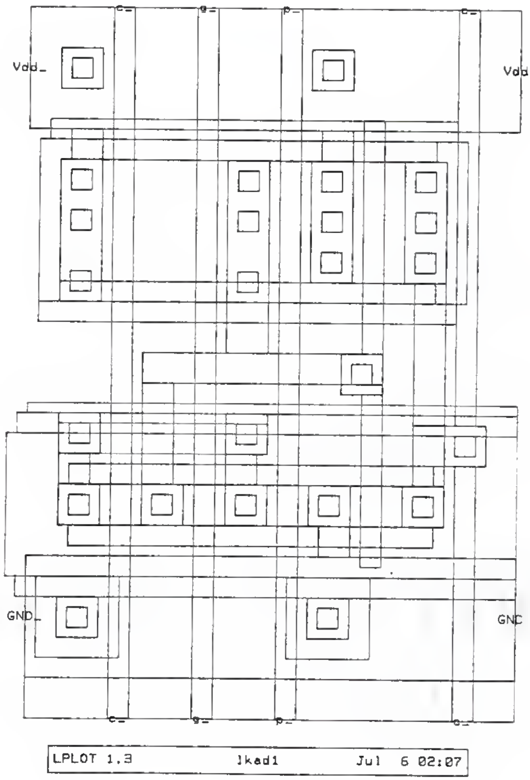

Figure A42: Composite Mask Layout For lkadl Cell

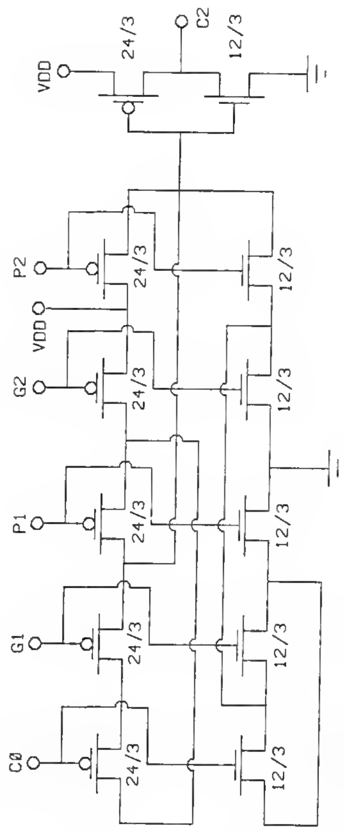

Figure A43: Transistor Schematic For The 1kad2 Cell

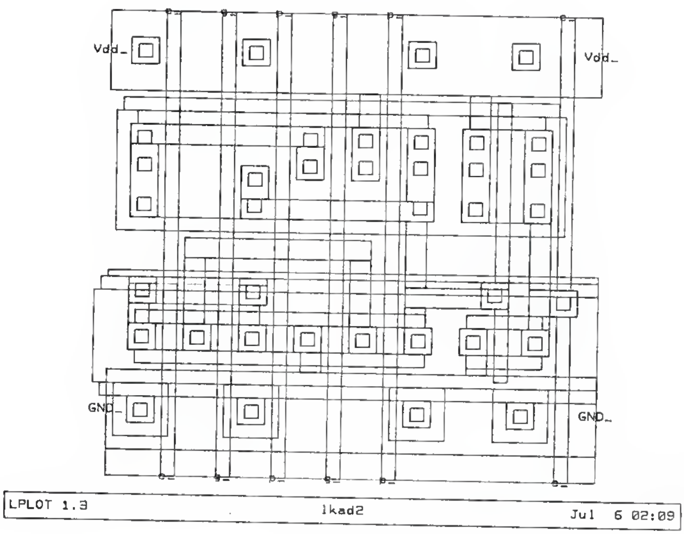

Figure A44: Composite Mask Layout For 1kad2 Cell

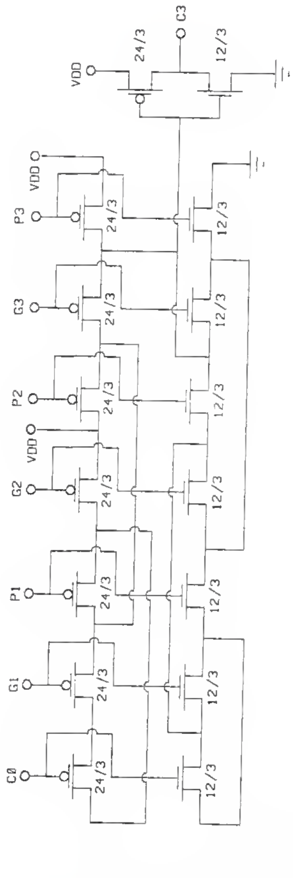

Figure A45: Transistor Schematic For The 1kad3 Cell

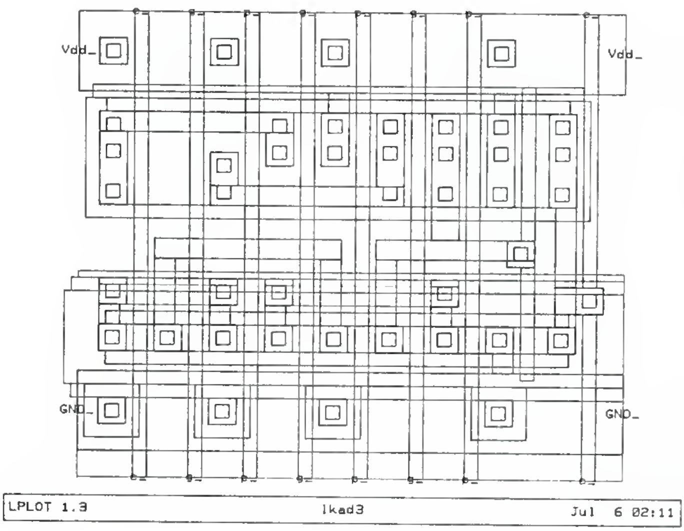

Figure A46: Composite Mask Layout For lkad3 Cell

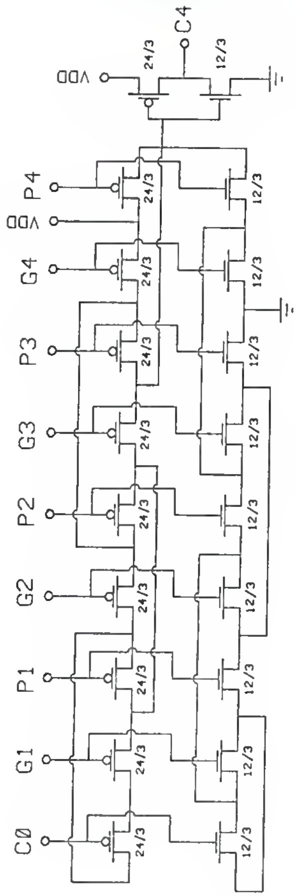

Figure A47: Transistor Schematic For The 1kad4 Cell

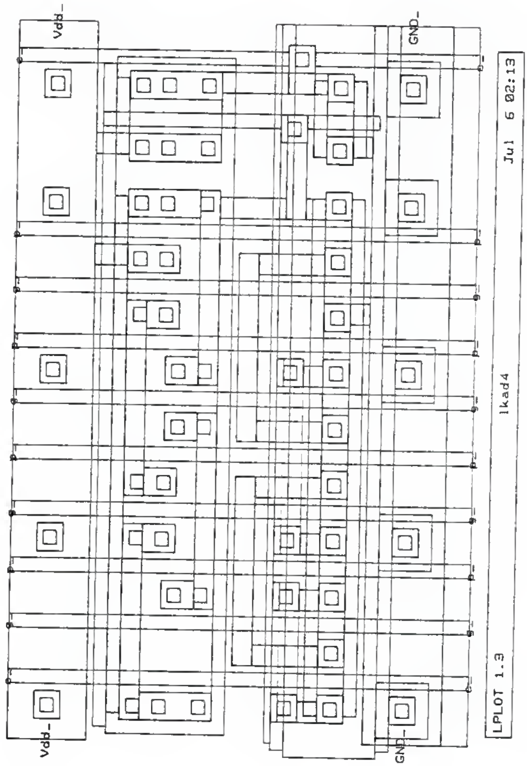

Figure A48: Composite Mask Layout For 1kad4 Cell

7.3 Flip-Flop Cells

For this design, two edge-triggered flip flops were necessary.

dffsr dreg

The first cell, a positive edge-triggered D flip-flop with set and reset inputs was constructed for use in the T phase clock and the program counter. The asynchronous reset is used in the TORO design to hardware reset the microprocessor to 0000. The second flip-flop was a positive edge-triggered D flip-flop with load inputs. This flip-flop was used for the register set and status register. Note from the transistor diagram for this flip-flop that buffers were included in the cell. These buffers must be inter-connected to the flip-flop when constructing routing for the macro that these flip-flops appear in. In addition, some feedback inter-connect from the D flip-flop output Q is necessary. Note the appropriate inter-connectivity to be used from the dotted lines in the dreg transistor schematic.

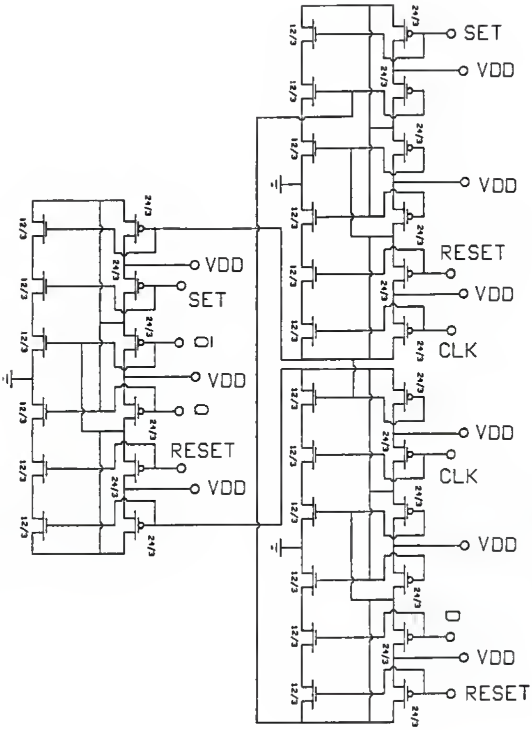

Figure A49: Transistor Schematic For The dffsr Cell

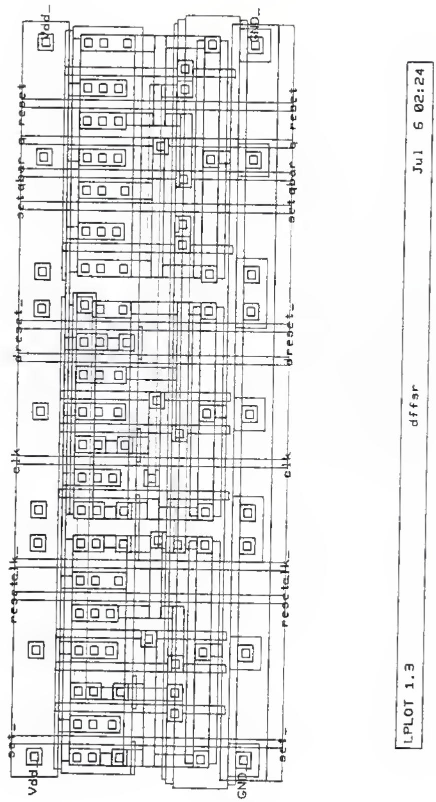

Figure A50: Composite Mask Layout For The dffsr Cell

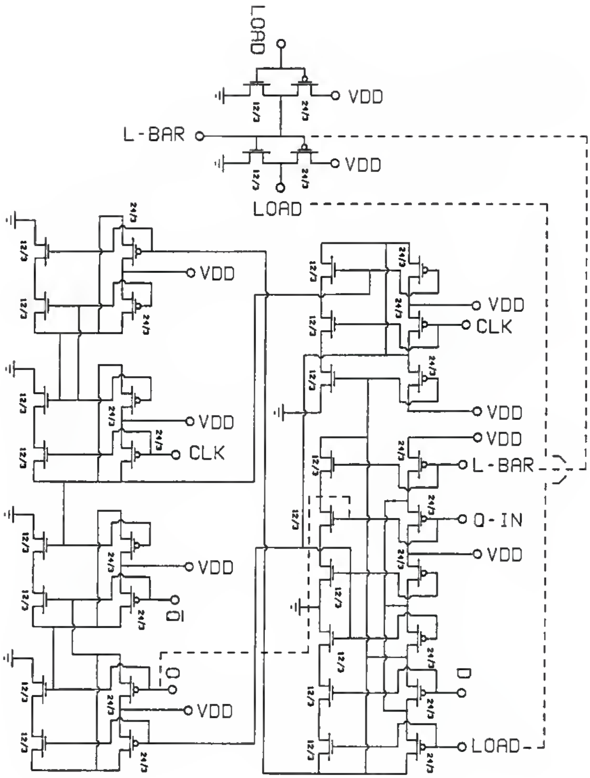

Figure A51: Transistor Schematic For The dreg Cell

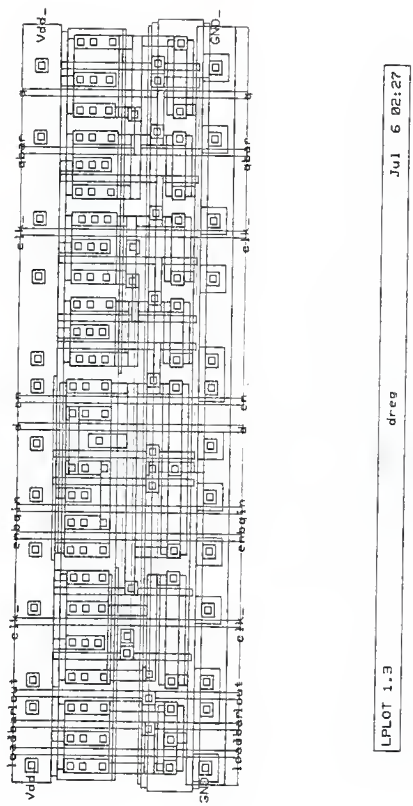

Figure A52: Composite Mask Layout For The dreg Cell

7.4 Pad Frame Cells

The cells for the pad frame were adapted from the MIT SCMOS 3-micron Pad Set available from MOSIS. The set includes an I/O pad, a Vdd pad, a Vss pad, and two corner pads.

KPIO	KPVDD	KPGND	Corn1	Corn2

In addition to the schematic diagram for the KPIO cell, a SPICE deck using worst case transistor parameters was also included. This deck was obtained using software from the North West Laboratory For Integrated Systems, and is part of the CAD package created at UC-Berkerley. Waveforms from simulation for the KPIO cell using the above SPICE deck gave propagation delays for the cell. The SPICE deck given simulated the OUT-pin-to-PAD propagation delay, three-state enabled. No schematics are given for the corner and power pad cells. They contain no transistors.

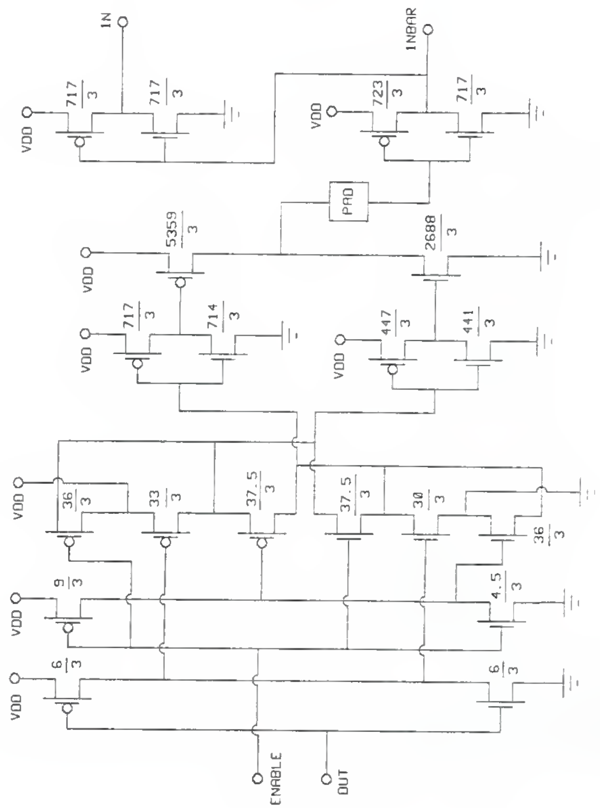

Figure A53: Transistor Schematic For The KPIO Cell

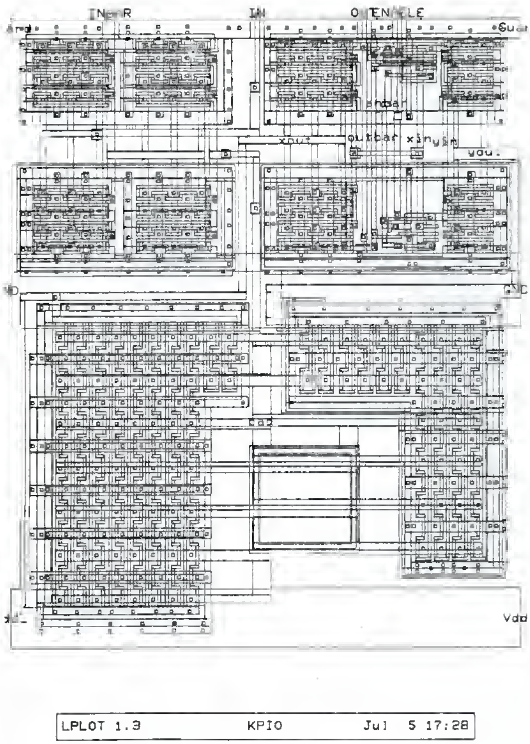

Figure A54: Composite Mask Layout For The KPIO Cell

```
KPIO
************************************************************
*                                                          *
*  This is a spice model for the KPIO pad by Joe           *
*  Varrientos for the TORO chip. The circuit was           *
*  extracted from MAGIC.                                   *
*                                                          *
*  This SPICE deck is to simulate delay for OUT-to-pad     *
*  for ENABLE high. Also, the rise/fall time is for        *
*  10ns.                                                   *
*                                                          *
************************************************************
*
*  Power Connections:
*
Vdd1 1 0 DC 5V
Vdd2 5 0 DC 5V
Vpsub 4 0 DC 5v
Vnsub 17 0 DC 0V
*
* THESE ARE WORST CASE VALUES FOR THE MOSIS 3u CMOS
* PROCESS
*
*
+ level=2.00          ld=0.320u           tox=550.e-10
+ nsub=1.0e+16        vto=1.000           kp=3.77e-05
+ gamma=1.50          phi=0.60            uo=600.0
+ uexp=1.000e-03      ucrit=999000.       delta=1.20
+ vmax=1.00e+05       xj=.600u            lambda=1.600e-02
+ nfs=1.200e+12       neff=1.00e-02       nss=0.0e+00
+ tpg=1.0             rsh=30              cgso=5.2e-10
+ cgdo=5.2e-10        cj=2.2e-4           mj=0.5
+ cjsw=3.0e-10        mjsw=0.33
*
+ level=2.00          ld=0.480u           tox=550.e-10
+ nsub=1.120e+14      vto=-1.00           kp=1.260e-05
+ gamma=0.700         phi=0.60            uo=200.0
+ uexp=0.150          ucrit=1.6e+04       delta=1.9
+ vmax=1.0e+05        xj=.400u            lambda=4.700e-02
+ nfs=8.800e+11       neff=1.00e-02       nss=0.0e+00
+ tpg=-1.0            rsh=70              cgso=4e-10
+ cgdo=4e-10          cj=3.5e-4           mj=0.5
+ cjsw=2.0e-10        mjsw=0.33
*
```

```
*
*  SPICE DECK created from KPIO.sim, tech=scmos
*
*
M1 7 6 5 4 CMOSP L=3.0U W=181.5U
M2 6 8 5 4 CMOSP L=3.0U W=183.0U
M3 7 6 5 4 CMOSP L=3.0U W=177.0U
M4 7 6 5 4 CMOSP L=3.0U W=181.5U
M5 6 8 5 4 CMOSP L=3.0U W=178.5U
M6 6 8 5 4 CMOSP L=3.0U W=183.0U
M7 7 6 5 4 CMOSP L=3.0U W=177.0U
M8 6 8 5 4 CMOSP L=3.0U W=178.5U
M9 10 9 5 4 CMOSP L=3.0U W=180.0U
M10 12 11 5 4 CMOSP L=3.0U W=6.0U
M11 10 9 5 4 CMOSP L=3.0U W=177.0U
M12 5 14 13 4 CMOSP L=3.0U W=9.0U
M13 5 14 15 4 CMOSP L=3.0U W=36.0U
M14 16 15 5 4 CMOSP L=3.0U W=112.5U
M15 16 15 5 4 CMOSP L=3.0U W=111.0U
M16 15 12 5 4 CMOSP L=3.0U W=33.0U
M17 9 13 15 4 CMOSP L=3.0U W=37.5U
M18 10 9 5 4 CMOSP L=3.0U W=181.5U
M19 16 15 5 4 CMOSP L=3.0U W=112.5U
M20 10 9 5 4 CMOSP L=3.0U W=178.5U
M21 16 15 5 4 CMOSP L=3.0U W=111.0U
M22 7 6 0 17 CMOSN L=3.0U W=180.0U
M23 6 8 0 17 CMOSN L=3.0U W=171.0U
M24 7 6 0 17 CMOSN L=3.0U W=178.5U
M25 6 8 0 17 CMOSN L=3.0U W=180.0U
M26 7 6 0 17 CMOSN L=3.0U W=180.0U
M27 6 8 0 17 CMOSN L=3.0U W=178.5U
M28 7 6 0 17 CMOSN L=3.0U W=178.5U
M29 6 8 0 17 CMOSN L=3.0U W=180.0U
M30 10 9 0 17 CMOSN L=3.0U W=168.0U
M31 16 15 0 17 CMOSN L=3.0U W=109.5U
M32 10 9 0 17 CMOSN L=3.0U W=168.0U
M33 16 15 0 17 CMOSN L=3.0U W=111.0U
M34 10 9 0 17 CMOSN L=3.0U W=168.0U
M35 9 14 15 17 CMOSN L=3.0U W=37.5U
M36 0 14 13 17 CMOSN L=3.0U W=4.5U
M37 10 9 0 17 CMOSN L=3.0U W=178.5U
M38 16 15 0 17 CMOSN L=3.0U W=109.5U
M39 0 12 9 17 CMOSN L=3.0U W=30.0U
M40 9 13 0 17 CMOSN L=3.0U W=36.0U
M41 16 15 0 17 CMOSN L=3.0U W=111.0U
M42 12 11 0 17 CMOSN L=3.0U W=6.0U
M43 8 10 1 4 CMOSP L=3.0U W=279.0U
M44 1 10 8 4 CMOSP L=3.0U W=351.0U
M45 8 10 1 4 CMOSP L=3.0U W=279.0U
```

```
M46 1 10 8 4 CMOSP L=3.0U W=351.0U
M47 8 10 1 4 CMOSP L=3.0U W=279.0U
M48 1 10 8 4 CMOSP L=3.0U W=358.5U
M49 8 10 1 4 CMOSP L=3.0U W=279.0U
M50 1 10 8 4 CMOSP L=3.0U W=351.0U
M51 8 10 1 4 CMOSP L=3.0U W=279.0U
M52 1 10 8 4 CMOSP L=3.0U W=351.0U
M53 8 10 1 4 CMOSP L=3.0U W=279.0U
M54 1 10 8 4 CMOSP L=3.0U W=351.0U
M55 8 10 1 4 CMOSP L=3.0U W=279.0U
M56 1 10 8 4 CMOSP L=3.0U W=351.0U
M57 8 10 1 4 CMOSP L=3.0U W=279.0U
M58 1 10 8 4 CMOSP L=3.0U W=351.0U
M59 8 10 1 4 CMOSP L=3.0U W=156.0U
M60 8 10 1 4 CMOSP L=3.0U W=156.0U
M61 8 16 0 17 CMOSN L=3.0U W=159.0U
M62 8 16 0 17 CMOSN L=3.0U W=153.0U
M63 8 16 0 17 CMOSN L=3.0U W=153.0U
M64 8 16 0 17 CMOSN L=3.0U W=153.0U
M65 8 16 0 17 CMOSN L=3.0U W=153.0U
M66 8 16 0 17 CMOSN L=3.0U W=153.0U
M67 8 16 0 17 CMOSN L=3.0U W=543.0U
M68 8 16 0 17 CMOSN L=3.0U W=537.0U
M69 8 16 0 17 CMOSN L=3.0U W=537.0U
M70 8 16 0 17 CMOSN L=3.0U W=537.0U
*
* This cap is for the pad:
*
*  C71 18 0 223.0F
*
C71 8 0 223.0F
C72 1 0 6130.0F
C73 16 0 3297.0F
C74 15 0 917.0F
C75 13 0 124.0F
C76 12 0 117.0F
C77 10 0 5551.0F
C78 9 0 1338.0F
C79 7 0 1017.0F
C80 8 0 7527.0F
C81 6 0 2224.0F
C82 5 0 5798.0F
*
*
```

```
*  The following capacitances are for input
*  and output capacitive loads.
*
*  Node 14 is ENABLE, Node 7 is IN, Node 8 is pad
*  Node 11 is OUT, Node 6 is INBAR
*
C90 14 0 2.0PF
C91 7 0 0.5PF
C92 8 0 50.0PF
C93 11 0 5.0PF
C94 6 0 0.5PF
*
*  Nodeset for intial DC analysis
*  and convergence:
*
*
*
*  Simulation Parameters:
*
Vout 11 0 PULSE(0,5,10ns,10ns,10ns,50ns,100ns)
Venable 14 0 5V
*
*  Cards:
*
 .TRAN 1ns 150ns
 .END
```

```
---------------------------------------------------------------
                        KPIO
Legend:  + = v(11)               * = v(8)
---------------------------------------------------------------
TIME      -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
-----------|----|----|----|----|----|----|----|
 0.000e+00 .    X    .    .    .    .    .    .
 8.242e-10 .   *+    .    .    .    .    .    .
 1.648e-09 .   *+    .    .    .    .    .    .
 2.473e-09 .   *+    .    .    .    .    .    .
 3.297e-09 .   *+    .    .    .    .    .    .
 4.121e-09 .    X    .    .    .    .    .    .
 4.945e-09 .    X    .    .    .    .    .    .
 5.769e-09 .    X    .    .    .    .    .    .
 6.593e-09 .    X    .    .    .    .    .    .
 7.418e-09 .    X    .    .    .    .    .    .
 8.242e-09 .    X    .    .    .    .    .    .
 9.066e-09 .    X    .    .    .    .    .    .
 9.890e-09 .    X    .    .    .    .    .    .
 1.071e-08 .   *.+   .    .    .    .    .    .
 1.154e-08 .   *.  + .    .    .    .    .    .
 1.236e-08 .   *.    +    .    .    .    .    .
 1.319e-08 .   *.    . +  .    .    .    .    .
 1.401e-08 .   *.    .    +    .    .    .    .
 1.484e-08 .   *.    .    . +  .    .    .    .
 1.566e-08 .   *.    .    .   +.    .    .    .
 1.648e-08 .   *.    .    .    .+   .    .    .
 1.731e-08 .    *    .    .    .  + .    .    .
 1.813e-08 .    *    .    .    .    +    .    .
 1.896e-08 .    *    .    .    .    . +  .    .
 1.978e-08 .    *    .    .    .    .   +.    .
 2.060e-08 .    *    .    .    .    .    +    .
 2.143e-08 .    *    .    .    .    .    +    .
 2.225e-08 .    *    .    .    .    .    +    .
 2.308e-08 .    *    .    .    .    .    +    .
 2.390e-08 .    *    .    .    .    .    +    .
 2.473e-08 .   *.    .    .    .    .    +    .
 2.555e-08 .   *.    .    .    .    .    +    .
 2.637e-08 .   *.    .    .    .    .    +    .
 2.720e-08 .   *.    .    .    .    .    +    .
 2.802e-08 .   *.    .    .    .    .    +    .
 2.885e-08 .   *.    .    .    .    .    +    .
 2.967e-08 .   *.    .    .    .    .    +    .
 3.049e-08 .   *.

 3.214e-08 .   *.    .    .    .    .    +    .
-----------|----|----|----|----|----|----|----|
TIME      -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
```

```
TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
-----------|----|----|----|----|----|----|----|
 3.297e-08 .   *.    .    .    .    .    +    .
 3.379e-08 .   *.    .    .    .    .    +    .
 3.462e-08 .   *.    .    .    .    .    +    .
 3.544e-08 .   *.    .    .    .    .    +    .
 3.626e-08 .   *.    .    .    .    .    +    .
 3.709e-08 .   *.    .    .    .    .    +    .
 3.791e-08 .   *.    .    .    .    .    +    .
 3.874e-08 .   *.    .    .    .    .    +    .
 3.956e-08 .   *.    .    .    .    .    +    .
 4.038e-08 .    *    .    .    .    .    +    .
 4.121e-08 .    *    .    .    .    .    +    .
 4.203e-08 .    *    .    .    .    .    +    .
 4.286e-08 .    *    .    .    .    .    +    .
 4.368e-08 .    .*   .    .    .    .    +    .
 4.451e-08 .    . *  .    .    .    .    +    .
 4.533e-08 .    .  * .    .    .    .    +    .
 4.615e-08 .    .    *    .    .    .    +    .
 4              .    .*   .    .    .    +    .
 4.780e-08 .    .    .  * .    .    .    +    .
 4.863e-08 .    .    .    .*   .    .    +    .
 4.945e-08 .    .    .    .   *.    .    +    .
 5.027e-08 .    .    .    .    . *  .    +    .
 5.110e-08 .    .    .    .    .   *.    +    .
 5.192e-08 .    .    .    .    .    .*   +    .
 5.275e-08 .    .    .    .    .    . *  +    .
 5.357e-08 .    .    .    .    .    .  * +    .
 5.440e-08 .    .    .    .    .    .   *+    .
 5.522e-08 .    .    .    .    .    .   *+    .
 5.604e-08 .    .    .    .    .    .   *+    .
 5.687e-08 .    .    .    .    .    .   *+    .
 5.769e-08 .    .    .    .    .    .   *+    .
 5.852e-08 .    .    .    .    .    .   *+    .
 5.934e-08 .    .    .    .    .    .   *+    .
 6.016e-08 .    .    .    .    .    .   *+    .
 6.099e-08 .    .    .    .    .    .   *+    .
 6.181e-08 .    .    .    .    .    .   *+    .
 6.264e-08 .    .    .    .    .    .   *+    .
 6.346e-08 .    .    .    .    .    .   *+    .
 6.429e-08 .    .    .    .    .    .   *+    .
 6.511e-08 .    .    .    .    .    .   *+    .
 6.593e-08 .    .    .    .    .    .   *+    .
 6.676e-08 .    .    .    .    .    .   *+    .
 6.758e-08 .    .    .    .    .    .   *+    .
-----------|----|----|----|----|----|----|----|
TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
```

```
 TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
-----------|----|----|----|----|----|----|----|
 6.841e-08 .    .    .    .    .    .   *+    .
 6.923e-08 .    .    .    .    .    .   *+    .
 7.005e-08 .    .    .    .    .    .   X.    .
 7.088e-08 .    .    .    .    .    . + *.    .
 7.170e-08 .    .    .    .    .    +   =  . .
 7.253e-08 .    .    .    .    .  + .   *.    .
 7.335e-08 .    .    .    .    .+   .   *.    .
 7.418e-08 .    .    .    .   +.    .   *.    .
 7.500e-08 .    .    .    . +  .    .   *.    .
 7.582e-08 .    .    .    +    .    .   *.    .
 7.665e-08 .    .    .  + .    .    .   *.    .
 7.747e-08 .    .    .+   .    .    .   *.    .
 7.830e-08 .    .   +.    .    .    .   *.    .
 7.912e-08 .    . +  .    .    .    .   *.    .
 7.995e-08 .    +    .    .    .    .   *.    .
 8.077e-08 .    +    .    .    .    .   *.    .
 8.159e-08 .    +    .    .    .    .   *.    .
 8.242e-08 .    +    .    .    .    .    *    .
 8.324e-08 .    +    .    .    .    .    *    .
 8.407e-08 .    +    .    .    .    .    *    .
 8.489e-08 .    +    .    .    .    .    *    .
 8.571e-08 .    +    .    .    .    .    *    .
 8.654e-08 .    +    .    .    .    .    *    .
 8.736e-08 .    +    .    .    .    .    *    .
 8.819e-08 .    +    .    .    .    .    *    .
 8.901e-08 .    +    .    .    .    .    *    .
 8.984e-08 .    +    .    .    .    .    *    .
 9.066e-08 .    +    .    .    .    .    *    .
 9.148e-08 .    +    .    .    .    .    *    .
 9.231e-08 .    +    .    .    .    .    *    .
 9.313e-08 .    +    .    .    .    .    *    .
 9.396e-08 .    +    .    .    .    .    *    .
 9.478e-08 .    +    .    .    .    .    *    .
 9.560e-08 .    +    .    .    .    .    *    .
 9.643e-08 .    +    .    .    .    .    *    .
 9.725e-08 .    +    .    .    .    .   *.    .
 9.808e-08 .    +    .    .    .    . * .    .
 9.890e-08 .    +    .    .    .   *.    .    .
 9.973e-08 .    +    .    .    *    .    .    .

 1.014e-07 .    +    *    .    .    .    .    .
 1.022e-07 .    + *  .    .    .    .    .    .
 1.030e-07 .    +*   .    .    .    .    .    .
-----------|----|----|----|----|----|----|----|
 TIME       -1.00e0.00e1.00e2.00e3.00e4.00e5.00e6.00e+00
```

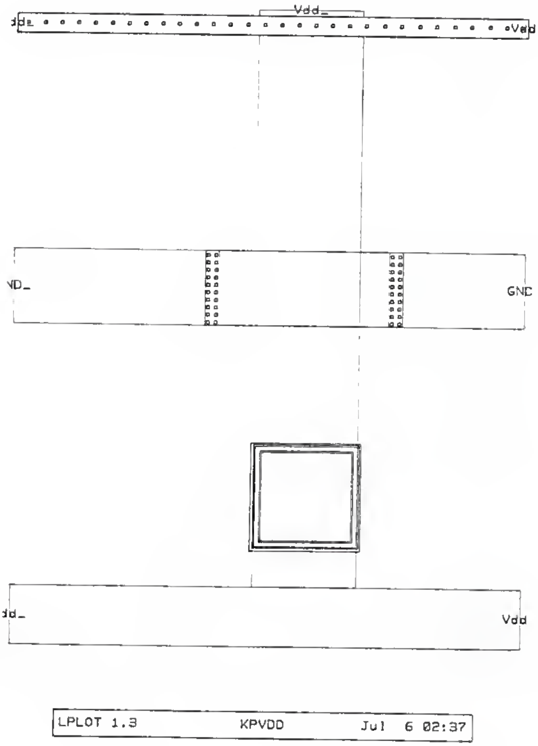

Figure A55: Composite Mask Layout For The KPVDD Cell

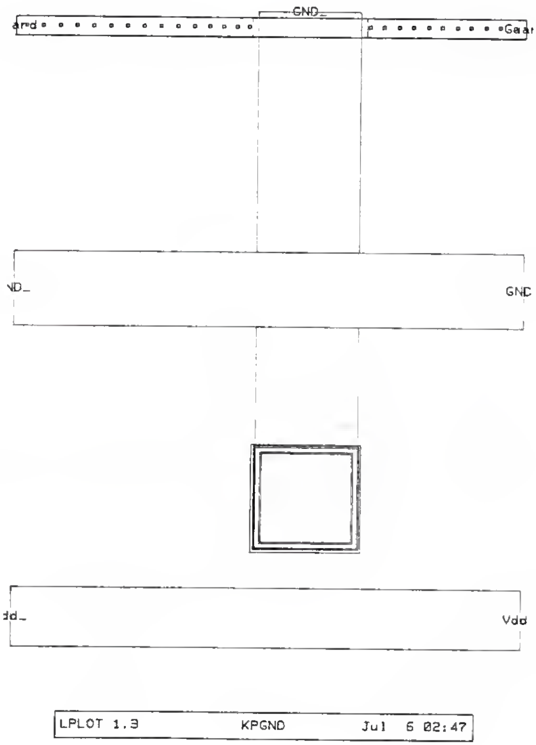

Figure A56: Composite Mask Layout For The KPGND Cell

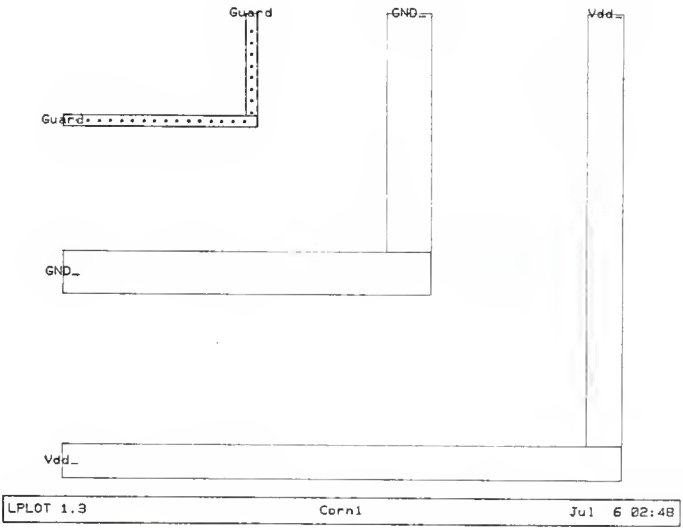

Figure A57: Composite Mask Layout For The Corn1 Cell

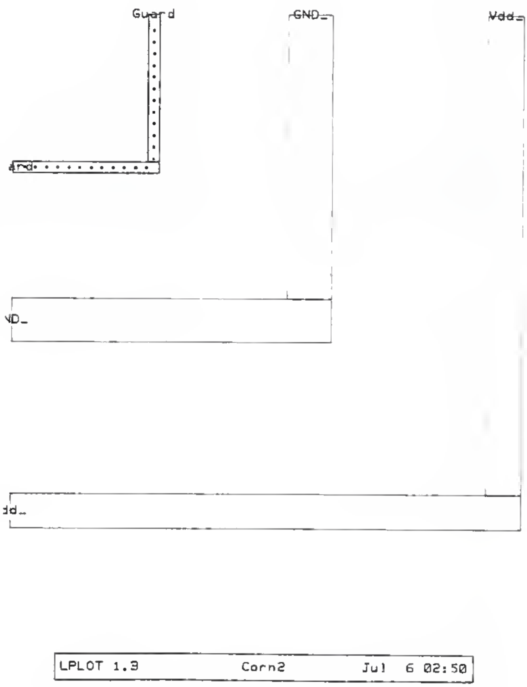

Figure A58: Composite Mask Layout For The Corn2 Cell

8.0 Appendix B - Design Simulation Library

The command files and plots for the simulations performed on the functional blocks of the TORO, as well as the final design, appear on the following pages. This appendix is divided into five major sections: one for each functional block, and one for the final design. All simulation that appear in this appendix were performed using the VIVID FACTS simulator. Thus, the simulation files given here are command files for FACTS. Some explanation is given in each file as to the meaning of the commands and their uses. A more detailed description of the commands can be found in the VIVID Designer's Documentation[24].

8.1 Register Set Library

Four simulations were performed for the register set. Recall that the register set contains the memory address register (MAR), the instruction register (IR), the temporary register (TMP), the accumulator (A), and the index register (X). In addition, it contains a 2-to-1 multiplexer and some simple decode for register loading. The four simulations performed the following functions:

RS.1ST: This simulation showed that the MAR, IR, and TMP could be loaded independently of one another, and that the registers would only be loaded on the rising edge of the system clock.

RS.2ND: This simulation showed that the bits of the MAR, IR, and TMP were each mutually exclusive, that is to say, that no two data inputs were shorted together such that errors could occur during register loading.

RS.3RD: This simulation showed that the A and X registers could be loaded independently as well, and that only one or the other could appear on the main internal bus via the 2-to-1 multiplexer.

RS.4TH: This simulation showed that the A and X registers were effectively isolated from the main internal bus by the three-state buffers attached to outputs of the 2-to-1 multiplexer.

These four tests were all that were performed on the register set. They all showed the functionality desired for the TORO680-16 register set. The simulation files and resulting plots appear on the following pages.

```
***********************************************************
*                                                         *
*     Simulation file for checking RS register set.       *
*                         RS.1ST                          *
*      This test checks for independent loading of        *
*             the TMP, IR and MAR registers.              *
*                                                         *
***********************************************************
*
*  This is after the overhaul
*  on the dflops to dregs.
*
***********************************************************
*                                                         *
*  The system clock:                                      *
*                                                         *
***********************************************************
*
cl clock 1000ns   0011001100110011001100110011001100110011
as clock clk1 clk2 clk3 clk4 clk5
as clock clk6 clk7 clk8 clk9 clk10
pl clk1
*
***********************************************************
*                                                         *
*  Data becomes available on the main bus:                *
*                                                         *
***********************************************************
*
cl mainbus 1000ns 0000111111111111100000000000011111111111
as mainbus mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
*
***********************************************************
*                                                         *
*  Clocks for loading the registers:                      *
*                                                         *
***********************************************************
*
cl ldir 1000ns    1110011100000000011100000000011100000001
as ldir ldir
cl ldmr 1000ns    1110000001110000000001110000000001110000
as ldmr ldmr
cl ldtp 1000ns    1110000000000111000000000111000000000111
as ldtp ldtp
*
```

```
*********************************************************
*                                                       *
*  Plotting the buses:                                  *
*                                                       *
*********************************************************
*
pl ir0 ir1 ir2 ir3 ir4 ir5 ir6 ir7
pl ldir
pl adr0 adr1 adr2 adr3 adr4 adr5 adr6 adr7
pl ldmr
pl tp0 tp1 tp2 tp3 tp4 tp5 tp6 tp7
pl ldtp
*
*********************************************************
*                                                       *
*  The ACC and INDEX registers are                      *
*  for another simulation.                              *
*                                                       *
*********************************************************
*
lo ldr0 ldr1 slr0 slr1 slr2
hi slrb
*
*********************************************************
*                                                       *
*  Simulation parameters:                               *
*                                                       *
*  Plot Step:                          ps 10ns          *
*  Power Output? ( y = yes ):          po y             *
*  Simulation Length:                  sl 1000ns        *
*                                                       *
*********************************************************
*
sl 1000ns
ps 10ns
cm + hpr
ti RS.1ST
pf RS.out1
po y
*
*********************************************************
*                                                       *
*  Power given by FACTS after simulation:               *
*                                                       *
*  Average Power:          5.17686 milliwatts           *
*  Average Current:        1.03537 milliamps            *
*                                                       *
*********************************************************
```

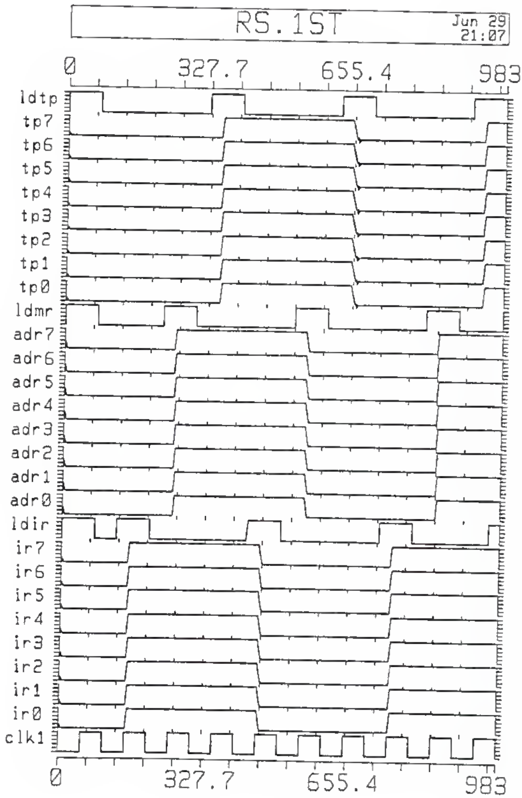

Figure B1: Plot Of Results From RS.1ST Simulation

```
************************************************************
*                                                          *
*       Simulation file for the RS register set.           *
*                          RS.2ND                          *
*      This simulation checks for independence of          *
*              the bits in the registers.                  *
*                                                          *
************************************************************
*
*  Updated 2/20/89 after overhauling the
*  dflops to dregs.
*
************************************************************
*                                                          *
*  System Clock:                                           *
*                                                          *
************************************************************
*
cl clock 1000ns   0011001100110011001100110011001100110011
as clock clk1 clk2 clk3 clk4 clk5
as clock clk6 clk7 clk8 clk9 clk10
pl clk1
*
************************************************************
*                                                          *
*  Data in from the main bus:                              *
*                                                          *
************************************************************
*
cl mbus 1000ns    0000111100001111000011110000111100001111
as mbus mb0 mb2 mb4 mb6
cl gbus 1000ns    1111000011110000111100001111000011110000
as gbus mb1 mb3 mb5 mb7
*
************************************************************
*                                                          *
*  Load the registers on every clock:                      *
*                                                          *
************************************************************
*
hi ldmr ldir ldtp slrb
lo ldr0 ldr1 slr0 slr1 slr2
*
```

```
*************************************************************
*                                                           *
* Plotting the buses:                                       *
*                                                           *
*************************************************************
*
pl ir0 ir1 ir2 ir3 ir4 ir5 ir6 ir7
pl adr0 adr1 adr2 adr3 adr4 adr5 adr6 adr7
pl tp0 tp1 tp2 tp3 tp4 tp5 tp6 tp7
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                            ps 10ns            *
*  Power Output? ( y = yes ):            po y               *
*  Simulation Length:                    sl 1000ns          *
*                                                           *
*************************************************************
*
sl 1000ns
ps 10ns
cm + hpr
pf RS.out2
ti RS.2ND
po y
*
*************************************************************
*                                                           *
*  Power given by FACTS after simulation:                   *
*                                                           *
*  Average Power:          6.10431 milliwatts               *
*  Average Current:        1.22086 milliamps                *
*                                                           *
*************************************************************
```

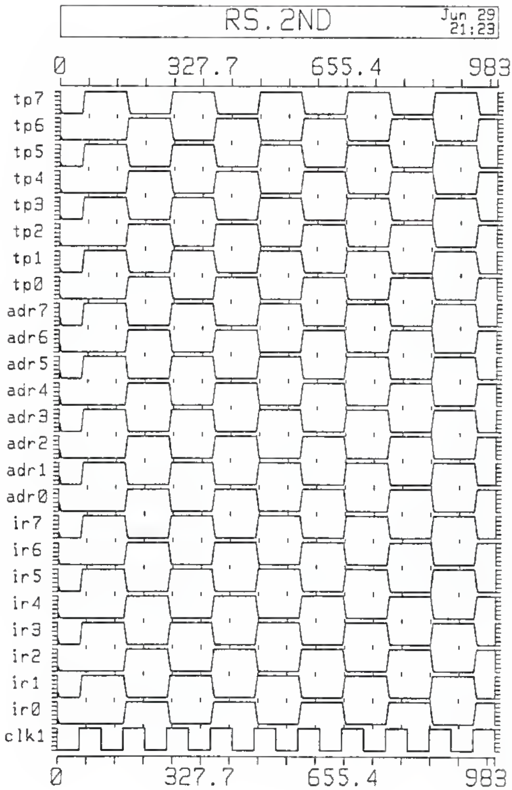

Figure B2: Plot Of Results From RS.2ND Simulation

```
*************************************************************
*                                                           *
*        Simulation file for the RS register set.           *
*                          RS.3RD                           *
*     This checks the loading of the A, X, and TMP.         *
*                                                           *
*************************************************************
*
*  Updated after overhaul of dflops to dregs.
*
*************************************************************
*                                                           *
*  System Clock:                                            *
*                                                           *
*************************************************************
*
cl clock 1000ns   0011001100110011001100110011001100110011
as clock clk1 clk2 clk3 clk4 clk5
as clock clk6 clk7 clk8 clk9 clk10
*
*************************************************************
*                                                           *
*  Clocking the main bus:                                   *
*                                                           *
*************************************************************
*
cl mb0 1000ns     0000111111111111100000000000011111111111
as mb0 mb0
cl mb1 1000ns     0000000011110000111111111111000011110000
as mb1 mb1
cl mb2 1000ns     1111111111110000000000001111111100000000
as mb2 mb2
cl mb3 1000ns     1111111100001111111100000000111111111111
as mb3 mb3
cl mb4 1000ns     1111000000000000000011110000000000001111
as mb4 mb4
cl mb5 1000ns     0000000000001111111111111111111100000000
as mb5 mb5
cl mb6 1000ns     1111000011110000111100001111000011110000
as mb6 mb6
cl mb7 1000ns     0000111111110000000000001111111100001111
as mb7 mb7
*
```

```
************************************************************
*                                                          *
*  Selecting the A and X registers. Here, the TMP is      *
*  always loaded:                                          *
*                                                          *
************************************************************
*
cl ldr 1000ns     1111111111111110000000000000000001111111
as ldr ldr0 ldr1
cl slr 1000ns     1111000011110000111100001111000011110000
as slr slr0 slr1 slr2
cl slrb 1000ns    0000111100001111000011110000111100001111
as slrb slrb
hi ldtp
*
************************************************************
*                                                          *
*  Plotting the outputs of the A, X and TMP:               *
*                                                          *
************************************************************
*
pl tp0 tp1 tp2 tp3 tp4 tp5 tp6 tp7
pl ax0 ax1 ax2 ax3 ax4 ax5 ax6 ax7
pl clk1
*
************************************************************
*                                                          *
*  Simulation Parameters:                                  *
*                                                          *
*  Plot Step:                            ps 10ns           *
*  Power Output? ( y = yes ):            po y              *
*  Simulation Length:                    sl 1000ns         *
*                                                          *
************************************************************
*
sl 1000ns
ps 10ns
cm + hpr
pf RS.out3
po y
cm + hpr
ti RS.3RD
*
```

```
************************************************************
*                                                          *
*  Power given from FACTS after simulation:                *
*                                                          *
*  Average Power:              5.53912 milliwatts          *
*  Average Current:            1.0782 milliamps            *
*                                                          *
************************************************************
```

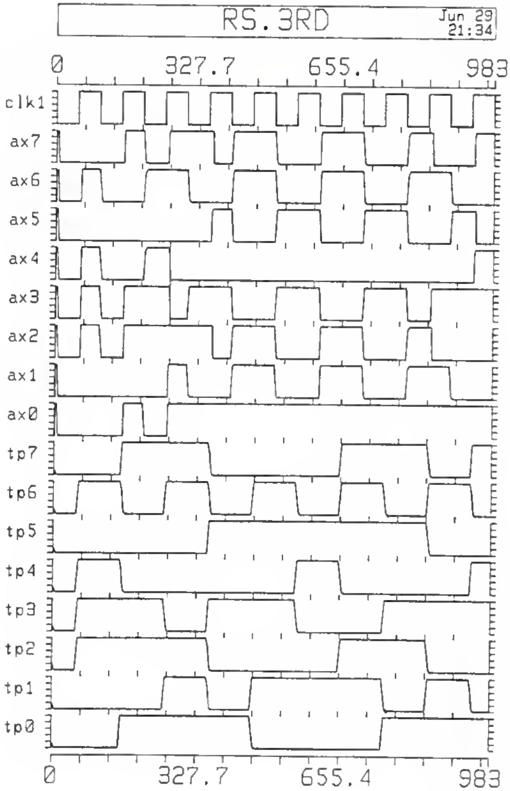

Figure B3: Plot Of Results From RS.3RD Simulation

```
***********************************************************
*                                                         *
*     Simulation file for testing the RS register.        *
*                        RS.4TH                           *
*   This checks the three-stating of the output of the    *
*                  A and X registers.                     *
*                                                         *
***********************************************************
*
*  set after overhaul of dflops to dregs.
*     2/20/89
*
***********************************************************
*                                                         *
*  Clocking the registers:                                *
*                                                         *
***********************************************************
*
cl clock 1000ns   0011001100110011001100110011001100110011
as clock clk1 clk2 clk3 clk4 clk5
as clock clk6 clk7 clk8 clk9 clk10
*
***********************************************************
*                                                         *
*  Here, the output bus from the AX multiplexer is        *
*  clocked:                                               *
*                                                         *
***********************************************************
*
cl ax0 1000ns    0000111111111111100000000000011111111111
as ax0 ax0
cl ax1 1000ns    0000000011110000111111111111000011110000
as ax1 ax1
cl ax2 1000ns    1111111111110000000000001111111100000000
as ax2 ax2
cl ax3 1000ns    1111111100001111111100000000111111111111
as ax3 ax3
*
***********************************************************
*                                                         *
*  Here the three-state control for the register is       *
*  clocked:                                               *
*                                                         *
***********************************************************
*
hi ldr0 ldr1
cl bsr 1000ns    1111111111111111000000000000000011111111
as bsr bsr1 bsr2
*
```

```
************************************************************
*                                                          *
*  Here the A and X registers are selected:                *
*                                                          *
************************************************************
*
cl slr 1000ns     1111000011110000111100001111000011110000
as slr slr0 slr1 slr2
cl slrb 1000ns    0000111100001111000011110000111100001111
as slrb slrb
*
************************************************************
*                                                          *
*  Plotting the buses:                                     *
*                                                          *
************************************************************
*
pl tst0 tst1 tst2 tst3
pl mb0 mb1 mb2 mb3
pl ax0 ax1 ax2 ax3
pl clk1
*
************************************************************
*                                                          *
*  Simulation Parameters:                                  *
*                                                          *
*  Plot Step:                          ps 10ns             *
*  Power Output? ( y = yes ):          po y                *
*  Simulation Length:                  sl 1000ns           *
*                                                          *
************************************************************
*
sl 1000ns
ps 10ns
cm + hpr
pf RS.out4
ti RS.4TH
po y
*
************************************************************
*                                                          *
*  Power given from FACTS after simulation:                *
*                                                          *
*  Average Power:                10.2483 milliwatts        *
*  Average Current:               2.04966 milliwatts       *
*                                                          *
************************************************************
```

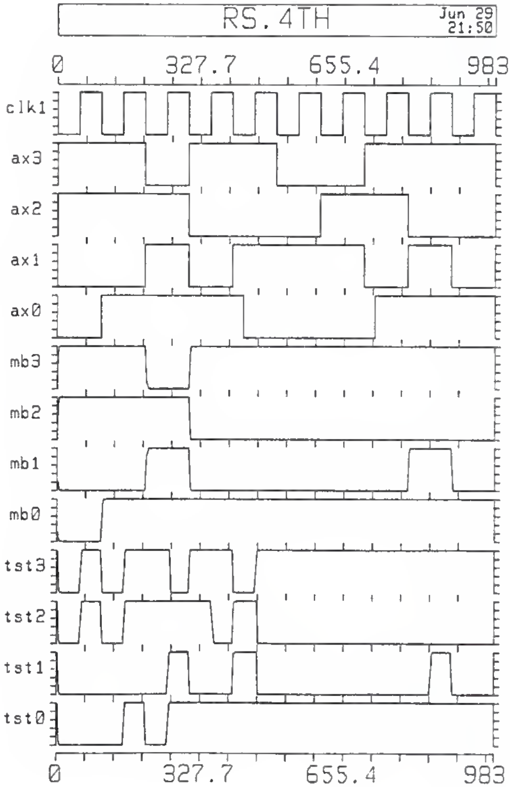

Figure B4: Plot Of Results From RS.4TH Simulation

8.2 Program Counter Library

Nine simulations were performed for the program counter. Recall that the program counter was constructed from four four-bit counter macros, each with an enable and ripple-carry out. In this set of tests, it was necessary to check each macro after final program counter construction to show that the counter could count from 0000 to FFFF without fail. It was also necessary to show that any value could be loaded into the counter. Note from the plots from the simulations that hazards appear for the ripple carry outs from the macros. These hazard occur at times soon after the rising edge of the system clock, and settle long before the rising edge of the next rising edge. Thus, no attempt was made to remove them. The following tests were performed:

PC.1ST: This simulation showed the functionality of the least significant nibble of the program counter during enable. It correctly enabled the next cascaded four-bit macro.

PC.2ND: This simulation showed the functionality of the next to least significant nibble of the program counter. For this simulation, the RCO out of the previous four-bit macro was forced high. The macro under simulation correctly enabled the next cascaded four-bit macro.

PC.3RD: This simulation showed the functionality of the next to most significant nibble of the program counter. Again, the RCO out of the previous four-bit macro was forced high. The macro under simulation correctly enabled the next cascaded four-bit macro.

PC.4TH: This simulation showed the functionality of the most significant nibble of the program counter. Again, the RCO out of the previous four-bit macro was forced high.

PC.5TH: This simulation showed that the program counter could be effectively loaded. Loading was performed such that the program counter, when allowed to count, would again correctly enable the macros of the counter. This simulation showed that the program counter could count from 0000 to FFFF.

PC.6TH: This simulation showed that the program counter could be enabled and disabled effectively. This was done by clocking the enable input INCPC and observing the data inputs of the flip-flops used. For enable, the data inputs were one more than the value at the counter output. While disabled, the data inputs and counter outputs were equal.

PC.7TH: This simulation showed that the counter could be loaded, and that the outputs of the counter were

effectively isolated from the main internal bus by the program counter three-state buffers. This was done by clocking the load and three-state enable for the counter and observing the counter output and the main internal bus.

PC.8TH: This simulation showed that the counter could be asynchronously reset independent of the system clock and load inputs. This was done by clocking the load input and allowing data from the data load inputs to appear at the data inputs of the flips flops in the counter. The reset was clocked during both reset high and low conditions. Recall that the reset input for the counter is active low.

PC.9TH: This simulation showed that the counter could be loaded from the main bus and, while the load control input was inactive, could increment the value at the counter output. This was done by allowing data to change on the main internal bus, then latching onto it with the system clock. The ability to load the data on the main data bus was observed at the data inputs of the flip flops.

These nine tests showed the program counter to functionally perform as needed for the TORO design. Some additional information can be derived from the plots as to counter set-up and hold time, FOR THE COMBINATIONAL PORTION OF THE DESIGN. Flip flop set-up and hold times would require separate simulations on those cells using SPICE. The plots and simulation command files for FACTS for the program counter appear on the following pages.

```
**********************************************************
*                                                        *
*          This test is for the program counter          *
*                        PC.SIM1                         *
*         to check the least significant nibble.         *
*                                                        *
**********************************************************
*
**********************************************************
*                                                        *
*  Here is the clock. It is running at 10 MHz.           *
*                                                        *
**********************************************************
*
cl clock 100ns 01
as clock clk1 clk2
*
**********************************************************
*                                                        *
*  Here is the reset, enable and three-state control:   *
*                                                        *
**********************************************************
*
hi incpc bspc2 bspc1
lo ldpc1 ldpc2
cl reset 2000ns 0111111111111111111111111111111111111111
as reset rspc1 rspc2
*
**********************************************************
*                                                        *
*  Simulation Parameters:                                *
*                                                        *
*  Plot Step:                      ps 10ns               *
*  Power Output? ( y = yes ):      po y                  *
*  Simulation Length:              sl 2000ns             *
*                                                        *
**********************************************************
*
sl 2000ns
ps 10ns
ti PC.1ST
pf PC.1st.out
po y
cm + hpr
*
```

```
*************************************************************
*                                                           *
*  Plotting the outputs and the ripple-carrys:              *
*                                                           *
*************************************************************
*
pl main0 main1 main2 main3 main4 main5 main6 main7
pl main8 main9 main10 main11 main12 main13 main14 main15
pl rco0 rco1 rco2 rco3 clk1
*
*************************************************************
*                                                           *
*  These are the current and power figures from this       *
*  simulation from FACTS:                                   *
*                                                           *
*   Average power:             2.90554 mW                   *
*   Average current:           0.581109 mA                  *
*                                                           *
*************************************************************
```

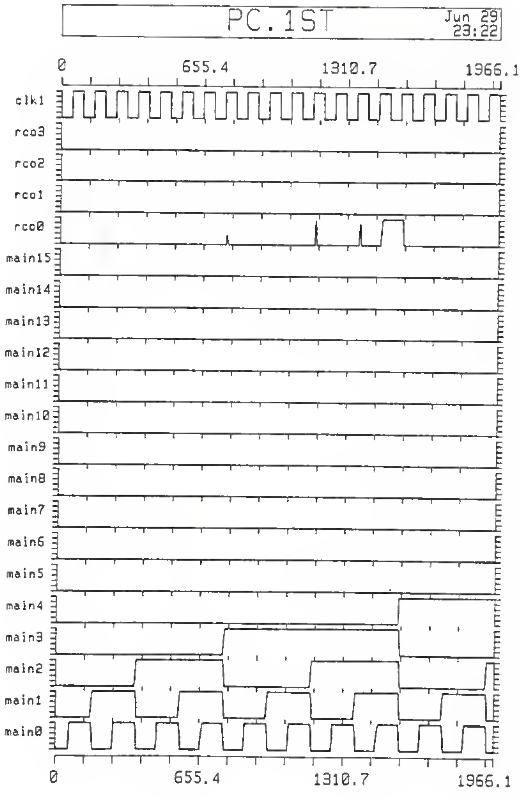

Figure B5: Plot of Results From PC.1ST Simulation

```
*************************************************************
*                                                           *
*          This test is for the program counter             *
*                         PC.SIM2                           *
*      to check the next to least significant nibble.      *
*                                                           *
*************************************************************
*
*************************************************************
*                                                           *
*  Here is the clock. It is running at 10 MHz.              *
*                                                           *
*************************************************************
*
cl clock 100ns 01
as clock clk1 clk2
*
*************************************************************
*                                                           *
*  Here is the reset, enable and three-state control:       *
*                                                           *
*************************************************************
*
hi rco0 bspc2 bspc1
lo ldpc1 ldpc2 incpc
cl reset 2000ns 0111111111111111111111111111111111111
as reset rspc1 rspc2
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                      ps 10ns                  *
*  Power Output? ( y = yes ):      po y                     *
*  Simulation Length:              sl 2000ns                *
*                                                           *
*************************************************************
*
sl 2000ns
ps 10ns
ti PC.2ND
pf PC.2nd.out
po y
cm + hpr
*
```

```
 ***********************************************************
*                                                          *
*  Plotting the outputs and the ripple-carrys:             *
*                                                          *
************************************************************
*
pl main0 main1 main2 main3 main4 main5 main6 main7
pl main8 main9 main10 main11 main12 main13 main14 main15
pl rco0 rco1 rco2 rco3 clk1
*
************************************************************
*                                                          *
*  These are the current and power figures from this       *
*  simulation from FACTS:                                  *
*                                                          *
*   Average power:              4.8907 mW                  *
*   Average current:            0.978141 mA                *
*                                                          *
************************************************************
```

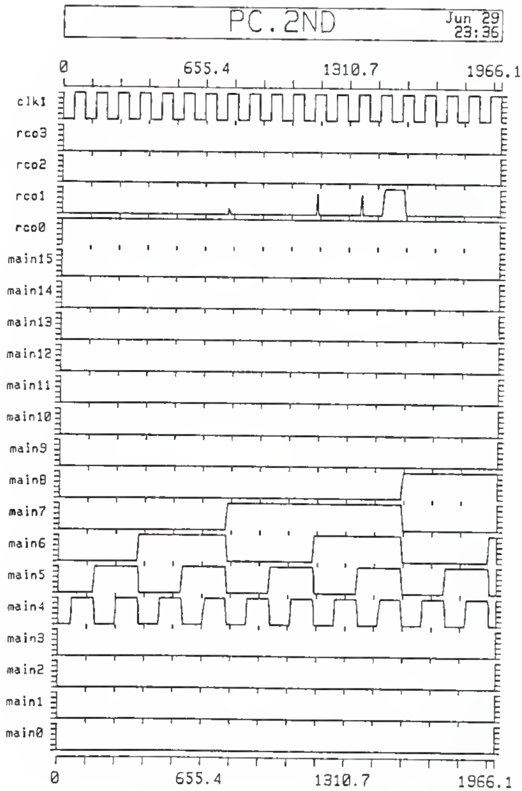

Figure B6: Plot of Results From PC.2ND Simulation

```
*********************************************************
*                                                       *
*         This test is for the program counter          *
*                        PC.SIM3                        *
*     to check the next to most significant nibble.     *
*                                                       *
*********************************************************
*
*********************************************************
*                                                       *
*  Here is the clock. It is running at 10 MHz.          *
*                                                       *
*********************************************************
*
cl clock 100ns 01
as clock clk1 clk2
*
*********************************************************
*                                                       *
*  Here is the reset, enable and three-state control:   *
*                                                       *
*********************************************************
*
hi rcol bspc2 bspc1
lo ldpc1 ldpc2 incpc
cl reset 2000ns 01111111111111111111111111111111111111111
as reset rspc1 rspc2
*
*********************************************************
*                                                       *
*  Simulation Parameters:                               *
*                                                       *
*  Plot Step:                        ps 10ns            *
*  Power Output? ( y = yes ):        po y               *
*  Simulation Length:                sl 2000ns          *
*                                                       *
*********************************************************
*
sl 2000ns
ps 10ns
ti PC.3RD
pf PC.3rd.out
po y
cm + hpr
*
```

```
**********************************************************
*                                                        *
*  Plotting the outputs and the ripple-carrys:           *
*                                                        *
**********************************************************
*
pl main0 main1 main2 main3 main4 main5 main6 main7
pl main8 main9 main10 main11 main12 main13 main14 main15
pl rco0 rco1 rco2 rco3 clk1
*
**********************************************************
*                                                        *
*  These are the current and power figures from this     *
*  simulation from FACTS:                                *
*                                                        *
*   Average power:            2.94544 mW                 *
*   Average current:          0.589089 mA                *
*                                                        *
**********************************************************
```

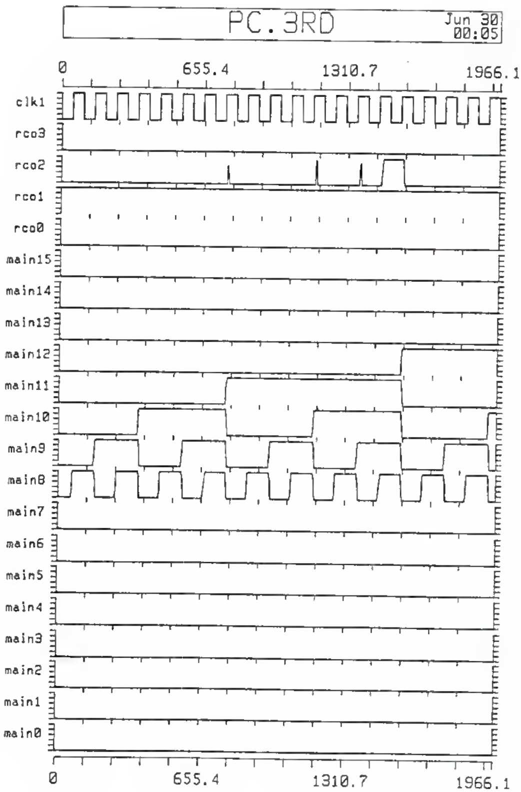

Figure B7: Plot Of Results From PC.3RD Simulation

```
***********************************************************
*                                                         *
*          This test is for the program counter           *
*                        PC.SIM4                          *
*      to check the next the most significant nibble.     *
*                                                         *
***********************************************************
*
***********************************************************
*                                                         *
*  Here is the clock. It is running at 10 MHz.            *
*                                                         *
***********************************************************
*
cl clock 100ns 01
as clock clk1 clk2
*
***********************************************************
*                                                         *
*  Here is the reset, enable and three-state control:     *
*                                                         *
***********************************************************
*
hi rco2 bspc2 bspc1
lo ldpc1 ldpc2 incpc
cl reset 2000ns 0111111111111111111111111111111111111
as reset rspc1 rspc2
*
***********************************************************
*                                                         *
*  Simulation Parameters:                                 *
*                                                         *
*  Plot Step:                       ps 10ns               *
*  Power Output? ( y = yes ):       po y                  *
*  Simulation Length:               sl 2000ns             *
*                                                         *
***********************************************************
*
sl 2000ns
ps 10ns
ti PC.4TH
pf PC.4th.out
po y
cm + hpr
*
```

```
************************************************************
*                                                          *
*  Plotting the outputs and the ripple-carrys:             *
*                                                          *
************************************************************
*
pl main0 main1 main2 main3 main4 main5 main6 main7
pl main8 main9 main10 main11 main12 main13 main14 main15
pl rco0 rco1 rco2 rco3 clk1
*
************************************************************
*                                                          *
*  These are the current and power figures from this      *
*  simulation from FACTS:                                  *
*                                                          *
*   Average power:               4.89957 mW                *
*   Average current:             0.979913 mA               *
*                                                          *
************************************************************
```

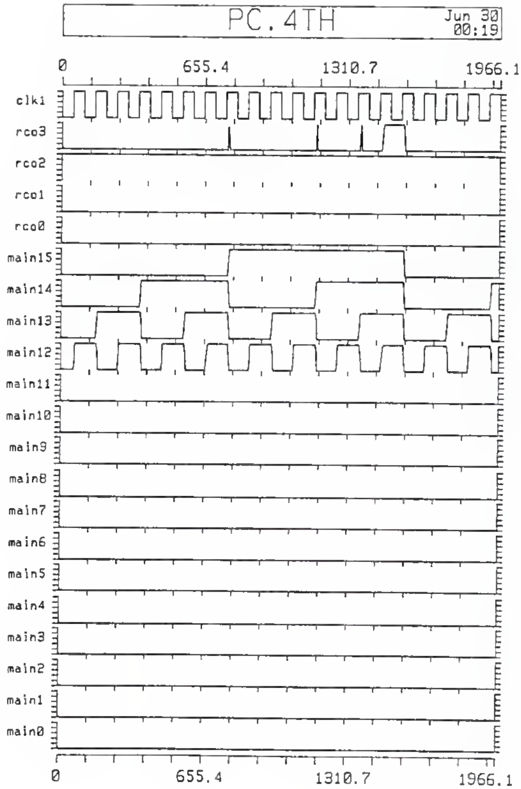

Figure B8: Plot Of Results From PC.4TH Simulation

```
**********************************************************
*                                                        *
*      Simulation file for checking Program Counter      *
*                          PC.SIM5                       *
*    Shows that program counter correctly increments     *
*      and enables/disables between nibble macros.       *
*                                                        *
**********************************************************
*
**********************************************************
*                                                        *
*  Enabling the counter and closing the 3-state          *
*  buffers:                                              *
*                                                        *
**********************************************************
*
hi incpc
lo bspc2 bspcl
*
*********************************************************
*                                                       *
*  Generating data on the main internal bus:            *
*                                                       *
*********************************************************
*
*
cl main0 2000ns  1111111111111111111111111111111111111111
as main0 main0
cl mainl 2000ns  1111111111111111111111111111111111111111
as mainl mainl
cl main2 2000ns  1111111111111111111111111111111111111111
as main2 main2
cl main3 2000ns  1111111111111111111111111111111111111111
as main3 main3
*
*
cl main4 2000ns  0000000000001111111111111111111111111111
as main4 main4
cl main5 2000ns  0000000000001111111111111111111111111111
as main5 main5
cl main6 2000ns  0000000000001111111111111111111111111111
as main6 main6
cl main7 2000ns  0000000000001111111111111111111111111111
as main7 main7
*
*
```

```
*
*
cl main8 2000ns  0000000000000000000011111111111111111111
as main8 main8
cl main9 2000ns  0000000000000000000011111111111111111111
as main9 main9
cl main10 2000ns 0000000000000000000011111111111111111111
as main10 main10
cl main11 2000ns 0000000000000000000011111111111111111111
as main11 main11
*
*
cl main12 2000ns 0000000000000000000000000000111111111111
as main12 main12
cl main13 2000ns 0000000000000000000000000000111111111111
as main13 main13
cl main14 2000ns 0000000000000000000000000000111111111111
as main14 main14
cl main15 2000ns 0000000000000000000000000000111111111111
as main15 main15
*
**********************************************************
*                                                        *
*  Clocking the load, reset, and clock inputs:           *
*                                                        *
**********************************************************
*
cl load  2000ns  1111000000001111000000001111000000001111
as load ldpc1 ldpc2
cl reset 2000ns  0111111111111111111111111111111111111111
as reset rspc1 rspc2
cl clock 2000ns  0011001100110011001100110011001100110011
as clock clk1 clk2
*
```

```
**********************************************************
*                                                        *
*  Simulation Parameters:                                *
*                                                        *
*  Plot Step:                         ps 10ns            *
*  Power Output? ( y = yes ):         po y               *
*  Simulation Length:                 sl 2000ns          *
*                                                        *
**********************************************************
*
sl 2000ns
ps 10ns
cm + hpr
pf PC.5th.out
ti PC.5TH
po y
*
**********************************************************
*                                                        *
*  Plotting the outputs of the flip-flops and the        *
*  ripple carry outs:                                    *
*                                                        *
**********************************************************
*
pl q0 q1 q2 q3 q4 q5 q6 q7
pl q8 q9 q10 q11 q12 q13 q14 q15
pl rco0 rco1 rco2 rco3 clk1
*
**********************************************************
*                                                        *
*  Power given from FACTS after simulation:              *
*                                                        *
*  Average Power:                     2.61277 mW         *
*  Average Current:                   0.522553 mA        *
*                                                        *
**********************************************************
```

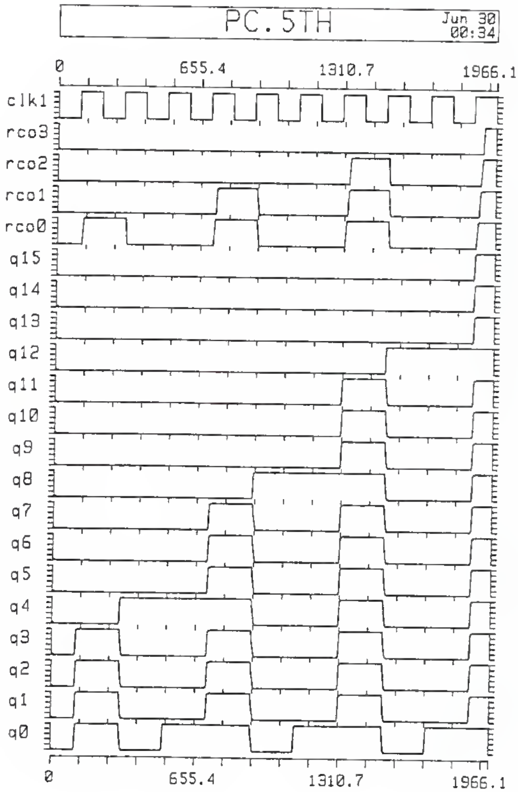

Figure B9: Plot Of Result From PC.5TH Simulation

```
************************************************************
*                                                          *
*       Simulation file for checking Program Counter      *
*                         PC.SIM6                          *
*        Shows that program counter correctly enables      *
*                    and disables.                         *
*                                                          *
************************************************************
*
************************************************************
*                                                          *
*   Enabling the counter and closing the 3-state           *
*   buffers:                                               *
*                                                          *
************************************************************
*
cl enable 2000ns 11111111000000001111000000001111111111111
as enable incpc
lo bspc2 bspc1
*
************************************************************
*                                                          *
*   Generating data on the main internal bus:              *
*                                                          *
************************************************************
*
*
cl main0 2000ns  1111111111111111111111111111111111111111
as main0 main0
cl main1 2000ns  1111111111111111111111111111111111111111
as main1 main1
cl main2 2000ns  1111111111111111111111111111111111111111
as main2 main2
cl main3 2000ns  1111111111111111111111111111111111111111
as main3 main3
*
*
cl main4 2000ns  1111111111111111111111111111111111111111
as main4 main4
cl main5 2000ns  1111111111111111111111111111111111111111
as main5 main5
cl main6 2000ns  1111111111111111111111111111111111111111
as main6 main6
cl main7 2000ns  1111111111111111111111111111111111111111
as main7 main7
*
*
```

```
*
*
cl main8 2000ns  0000000000000000000011111111111111111111
as main8 main8
cl main9 2000ns  0000000000000000000011111111111111111111
as main9 main9
cl main10 2000ns 0000000000000000000011111111111111111111
as main10 main10
cl main11 2000ns 0000000000000000000011111111111111111111
as main11 main11
*
*
cl main12 2000ns 0000000000000000000000000000001111111111
as main12 main12
cl main13 2000ns 0000000000000000000000000000001111111111
as main13 main13
cl main14 2000ns 0000000000000000000000000000001111111111
as main14 main14
cl main15 2000ns 0000000000000000000000000000001111111111
as main15 main15
*
**********************************************************
*                                                        *
*  Clocking the load, reset, and clock inputs:           *
*                                                        *
**********************************************************
*
cl load  2000ns  1111000000000000000000000000000000000000
as load ldpc1 ldpc2
cl reset 2000ns  0111111111111111111111111111111111111111
as reset rspc1 rspc2
cl clock 2000ns  0011001100110011001100110011001100110011
as clock clk1 clk2
*
```

```
**********************************************************
*                                                        *
*   Simulation Parameters:                               *
*                                                        *
*   Plot Step:                        ps 10ns            *
*   Power Output? ( y = yes ):        po y               *
*   Simulation Length:                sl 2000ns          *
*                                                        *
**********************************************************
*
sl 2000ns
ps 10ns
cm + hpr
pf PC.6th.out
ti PC.6TH
po y
*
**********************************************************
*                                                        *
*   Plotting the outputs of the flip-flops and the       *
*   ripple carry outs:                                   *
*                                                        *
**********************************************************
*
pl q0 q1 q2 q3 q4 q5 q6 q7
pl q8 q9 q10 q11 q12 q13 q14 q15
pl rco0 rco1 rco2 rco3 clk1
*
**********************************************************
*                                                        *
*   Power given from FACTS after simulation:             *
*                                                        *
*   Average Power:                 1.50007 mW            *
*   Average Current:               0.300013 mA           *
*                                                        *
**********************************************************
```

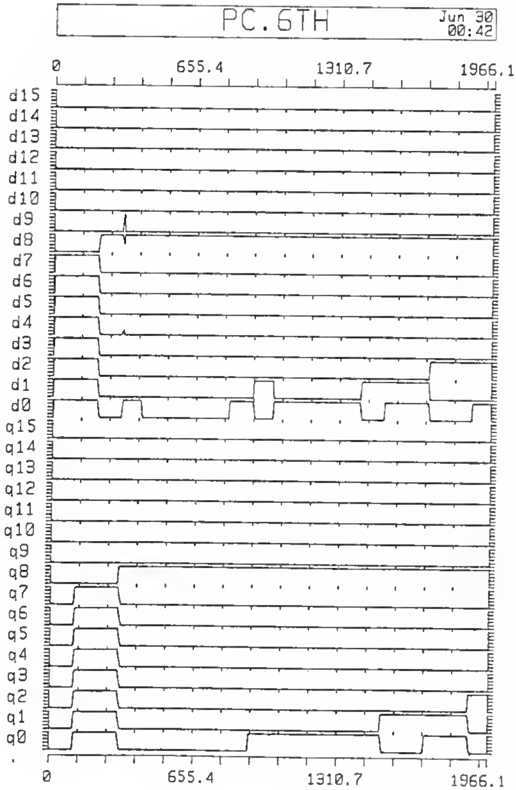

Figure B10: Plot Of Results From PC.6TH Simulation

```
************************************************************
*                                                          *
*    Simulation file for checking the Program Counter      *
*                          PC.7TH                          *
*     This simulation shows that the program counter       *
*          can be effectively isolated from the            *
*                  main internal bus.                      *
*                                                          *
************************************************************
*
************************************************************
*                                                          *
*  Clocking the enable inputs of the program               *
*  counter three-state buffers:                            *
*                                                          *
************************************************************
*
cl bus 2000ns    1111111100000000111111110000000011111111
as bus bspc2 bspc1
hi incpc
*
************************************************************
*                                                          *
*  Clocking the data inputs of the flip-flops:             *
*                                                          *
************************************************************
*
*
cl d0 2000ns     1100001111111100001111000000111111111100
as d0 d0
cl d1 2000ns     0000111111111100000000001111110000111111
as d1 d1
cl d2 2000ns     1111000000111111000011111111110000000000
as d2 d2
cl d3 2000ns     1111111111111111000000000000001111111111
as d3 d3
*
*
cl d4 2000ns     0000111100001111000011110000111100001111
as d4 d4
cl d5 2000ns     1111000011111111000011111111000011110000
as d5 d5
cl d6 2000ns     1111111111000000111111000000000000111111
as d6 d6
cl d7 2000ns     1111000011111111000011110000111111110000
as d7 d7
*
*
```

```
*
*
cl d8  2000ns       1111000000000000000000000000000000001111
as d8  d8
cl d9  2000ns       0000000000001111111111111111000000001111
as d9  d9
cl d10 2000ns       0000111111110000000000000000111111111111
as d10 d10
cl d11 2000ns       1111111111110000000011111111000000001111
as d11 d11
*
*
cl d12 2000ns       0000000011111111000000000000111111111111
as d12 d12
cl d13 2000ns       1111111111110000000011111111000000001111
as d13 d13
cl d14 2000ns       0000000000000000111111110000000011111111
as d14 d14
cl d15 2000ns       1111000011110000111100001111000000000000
as d15 d15
*
*************************************************************
*                                                           *
*  Enabling the load data input and resetting the           *
*  program counter and generating the clock:                *
*                                                           *
*************************************************************
*
cl load  2000ns  11
as load ldpc1 ldpc2
cl reset 2000ns  0111111111111111111111111111111111111111
as reset rspc1 rspc2
cl clock 2000ns  0011001100110011001100110011001100110011
as clock clk1 clk2
*
*************************************************************
*                                                           *
*  Plotting the counter outputs and the main                *
*  internal bus nodes:                                      *
*                                                           *
*************************************************************
*
pl q0 q1 q2 q3 q4 q5 q6 q7
pl q8 q9 q10 q11 q12 q13 q14 q15
pl main0 main1 main2 main3 main4 main5 main6 main7
pl main8 main9 main10 main11 main12 main13 main14 main15
*
```

```
**************************************************************
*                                                            *
*  Simulation Parameters:                                    *
*                                                            *
*  Plot Step:                          ps 10ns               *
*  Power Output? ( y = yes ):          po y                  *
*  Simulation Length:                  sl 2000ns             *
*                                                            *
**************************************************************
*
cm + hpr
ti PC.7TH
pf PC.out7
po y
sl 2000ns
ps 10ns
*
**************************************************************
*                                                            *
*  Power given from FACTS after simulation:                  *
*                                                            *
*  Average Power:        10.6972  milliwatts                 *
*  Average Current:       2.1394  milliamps                  *
*                                                            *
**************************************************************
```

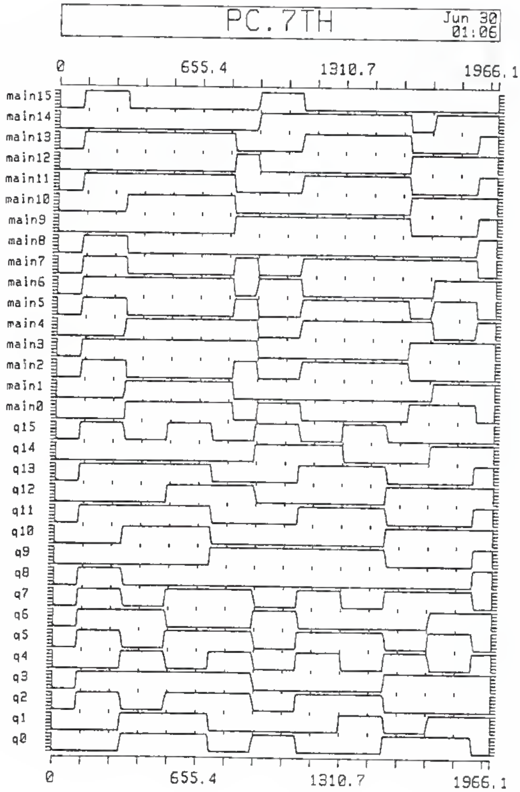

Figure B11: Plot Of Results From PC.7TH Simulation

```
**************************************************************
*                                                            *
*     Simulation file for checking Program Counter           *
*                         PC.SIM8                            *
*        Shows that program counter can be reset at          *
*            time, independent of the load input.            *
*                                                            *
**************************************************************
*
**************************************************************
*                                                            *
*  Enabling the counter and closing the 3-state              *
*  buffers:                                                  *
*                                                            *
**************************************************************
*
hi incpc
lo bspc2 bspc1
*
**************************************************************
*                                                            *
*  Generating data on the main internal bus:                 *
*                                                            *
**************************************************************
*
*
cl main0 2000ns  1111111111111111111111111111111111111111
as main0 main0
cl main1 2000ns  1111111111111111111111111111111111111111
as main1 main1
cl main2 2000ns  1111111111111111111111111111111111111111
as main2 main2
cl main3 2000ns  1111111111111111111111111111111111111111
as main3 main3
*
*
cl main4 2000ns  0000000000001111111111111111111111111111
as main4 main4
cl main5 2000ns  0000000000001111111111111111111111111111
as main5 main5
cl main6 2000ns  0000000000001111111111111111111111111111
as main6 main6
cl main7 2000ns  0000000000001111111111111111111111111111
as main7 main7
*
*
```

```
*
*
cl main8 2000ns  0000000000000000000011111111111111111111
as main8 main8
cl main9 2000ns  0000000000000000000011111111111111111111
as main9 main9
cl main10 2000ns 0000000000000000000011111111111111111111
as main10 main10
cl main11 2000ns 0000000000000000000011111111111111111111
as main11 main11
*
*
cl main12 2000ns 0000000000000000000000000000111111111111
as main12 main12
cl main13 2000ns 0000000000000000000000000000111111111111
as main13 main13
cl main14 2000ns 0000000000000000000000000000111111111111
as main14 main14
cl main15 2000ns 0000000000000000000000000000111111111111
as main15 main15
*
***********************************************************
*                                                         *
*  Clocking the load, reset, and clock inputs:            *
*                                                         *
***********************************************************
*
cl load  2000ns  1111000000001111000000001111000000001111
as load ldpc1 ldpc2
cl reset 2000ns  0111100011111111111111111000111111111111
as reset rspc1 rspc2
cl clock 2000ns  0011001100110011001100110011001100110011
as clock clk1 clk2
*
```

```
************************************************************
*                                                          *
*  Simulation Parameters:                                  *
*                                                          *
*  Plot Step:                        ps 10ns               *
*  Power Output? ( y = yes ):        po y                  *
*  Simulation Length:                sl 2000ns             *
*                                                          *
************************************************************
*
sl 2000ns
ps 10ns
cm + hpr
pf PC.out8
ti PC.8TH
po y
*
************************************************************
*                                                          *
*  Plotting the outputs of the flip-flops and the          *
*  ripple carry outs:                                      *
*                                                          *
************************************************************
*
pl q0 q1 q2 q3 q4 q5 q6 q7
pl q8 q9 q10 q11 q12 q13 q14 q15
pl rco0 rco1 rco2 rco3 clk1
*
************************************************************
*                                                          *
*  Power given from FACTS after simulation:                *
*                                                          *
*  Average Power:                Not Recorded              *
*  Average Current:              Not Recorded              *
*                                                          *
************************************************************
```

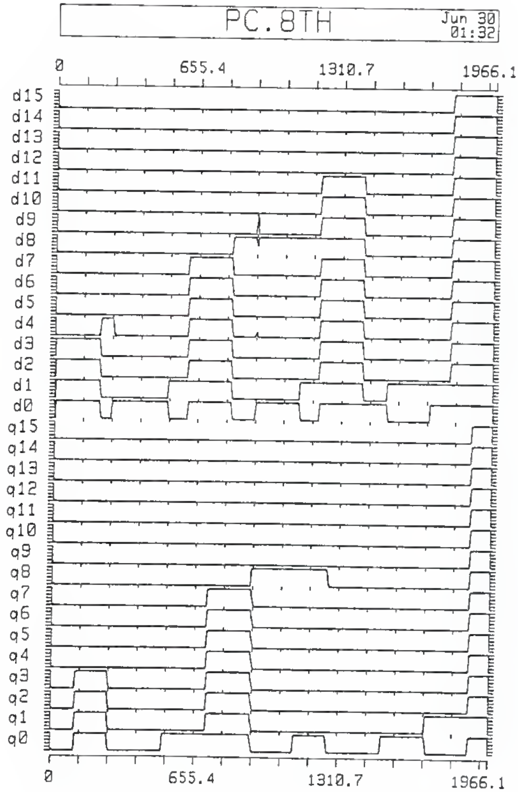

Figure B12: Plot Of Results From PC.8TH Simulation

```
*********************************************************
*                                                       *
*        Simulation for checking Program Counter.       *
*                        PC.9TH                         *
*  This simulation shows that the counter combinational *
*   logic loads and increments the counter correctly.   *
*                                                       *
*********************************************************
*
*********************************************************
*                                                       *
*  Enabling the counter and closing the three-state     *
*  buffers:                                             *
*                                                       *
*********************************************************
*
hi incpc
lo bspc2 bspcl
*
*********************************************************
*                                                       *
*  Clocking the main internal bus for loading the       *
*  counter:                                             *
*                                                       *
*********************************************************
*
cl main0 2000ns  10010010100100101001011110001100100111101
as main0 main0
cl mainl 2000ns  1001001001010010010010010010010010011110
as mainl mainl
cl main2 2000ns  1110011101011001000110010110010010010010
as main2 main2
cl main3 2000ns  1111100110010100010010100101111010010010
as main3 main3
*
*
cl main4 2000ns  0101000111001011010100010010100111101111
as main4 main4
cl main5 2000ns  1111111111100000000000100100010010111100
as main5 main5
cl main6 2000ns  1111001001111110001001011010001001000101 0
as main6 main6
cl main7 2000ns  1100011000100110110111110001110100101001
as main7 main7
*
*
```

```
*
*
cl main8 2000ns  0011001011001111111001110001100110100100
as main8 main8
cl main9 2000ns  0010010010010010111111111110101011011111
as main9 main9
cl main10 2000ns 0011101001011111110010010010010100100110
as main10 main10
cl main11 2000ns 1111110010010010101001000000111111001110
as main11 main11
*
*
cl main12 2000ns 0001000010010011111001001001010101010110
as main12 main12
cl main13 2000ns 0011111111001010100010001000101001000010
as main13 main13
cl main14 2000ns 0000010000100100111111111111110111011010
as main14 main14
cl main15 2000ns 0100100111111110001000100000000100000101 0
as main15 main15
*
*
************************************************************
*                                                          *
*  Clocking the load enable, system clock, and reset.      *
*                                                          *
************************************************************
*
cl load  2000ns  1110011001100110011001100110011001100110
as load ldpc1 ldpc2
cl reset 2000ns  0111111111111111111111111111111111111111
as reset rspc1 rspc2
cl clock 2000ns  0010001000100010001000100010001000100010
as clock clk1 clk2
*
************************************************************
*                                                          *
*  Plotting the flip flops data inputs and the             *
*  counter output:                                         *
*                                                          *
************************************************************
*
pl q0 q1 q2 q3 q4 q5 q6 q7
pl q8 q9 q10 q11 q12 q13 q14 q15
pl d0 d1 d2 d3 d4 d5 d6 d7
pl d8 d9 d10 d1 d12 d13 d14 d15
*
```

```
*************************************************************
*                                                           *
*   Simulation Parameters:                                  *
*                                                           *
*   Plot Step:                         ps 10ns              *
*   Power Output? ( y = yes ):         po y                 *
*   Simulation Length:                 sl 2000ns            *
*                                                           *
*************************************************************
*
sl 2000ns
ps 10ns
cm + hpr
pf PC.out9
ti PC.9TH
po y
*
*************************************************************
*                                                           *
*   Power given from FACTS after simulation:                *
*                                                           *
*   Average Power:                  4.93421 milliwatts      *
*   Average Current:                0.986842 milliwatts     *
*                                                           *
*************************************************************
```

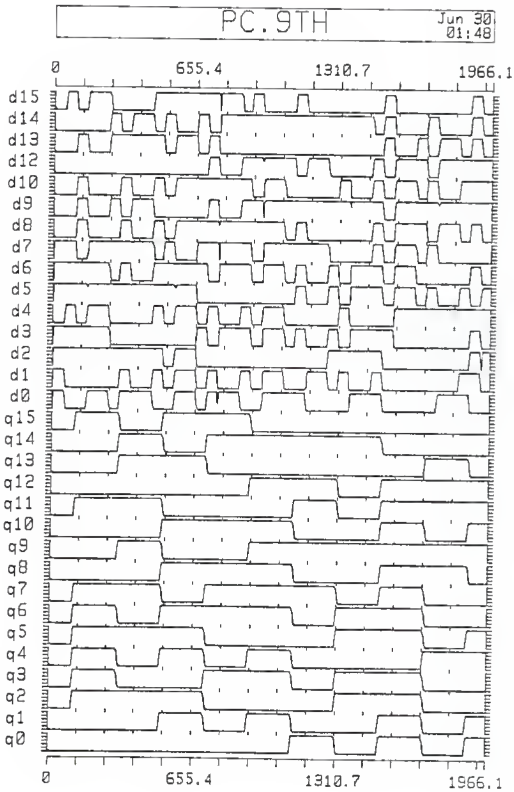

Figure B13: Plot Of Results From PC.9TH Simulation

8.3 Arithmetic Unit Library

Twelve tests were performed on the TORO arithmetic logic unit. Eleven were performed to show that the ALU could perform the data manipulation and operations necessary for the TORO design. One simulation was performed to show the effectiveness of the ALU three-state buffers. Below is a list of the simulations and a brief description of each. For each of the instructions, the TMP and AX inputs of the ALU, along with the input for the carry bit, were given in command files for FACTS. Control inputs were held static during each simulation.

ALU.AND: Simulation to show the functionality of the ALU during the AND instruction.

ALU.ADD: Simulation to show the functionality of the ALU during the ADD instruction.

ALU.OR: Simulation to show the functionality of the ALU during the OR instruction.

ALU.XOR: Simulation to show the functionality of the ALU during the exclusive-OR instruction.

ALU.CMP: Simulation to show the functionality of the ALU during the COMPARE instruction.

ALU.SHR: Simulation to show the functionality of the ALU during the SHIFT RIGHT instruction.

ALU.SHL: Simulation to show the functionality of the ALU during the SHIFT LEFT instruction.

ALU.INC: Simulation to show the functionality of the ALU during the INCREMENT instruction.

ALU.DEC: Simulation to show the functionality of the ALU during the DECREMENT instruction.

ALU.TST: Simulation to show the functionality of the ALU during the TEST instruction.

ALU.COM: Simulation to show the functionality of the ALU during the COMPLEMENT instruction.

ALU.BUS: Simulation to show that the output of the ALU could be effectively isolated from the main internal bus.

These simulations were sufficient to show the functionality of the ALU, but were by no means exhaustive. Computer time nor memory space could be made available for such testing. However, boundary conditions were tests. For example, FFFF was incremented and 0000 was decremented to test for correct carry propagation and carry bit generation. To this degree, the tests were complete. The plots and simulation command files for the above ALU tests appear on the following pages.

```
**********************************************************
*                                                        *
*          Simulation file for checking ALU.             *
*                       ALU.AND                          *
*           This is for the AND instruction.             *
*                                                        *
**********************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89.
*
**********************************************************
*                                                        *
*  First, the multiplexing control. The select alu       *
*  control signals are for the 4x1 mux at the output     *
*  of the alu. Used to perform shifts and rolls.         *
*                                                        *
**********************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
hi s01 s02 sb10 sb11
lo s11 s12 sb00 sb01
*
**********************************************************
*                                                        *
*  Control inputs for AX and TMP multiplexing. For this  *
*  instruction, the AX and TMP inputs are ANDed          *
*  together, then allowed to pass through the adder      *
*  and output multiplexer.                               *
*                                                        *
**********************************************************
*
hi tzro
lo tone aone szro
*
```

```
************************************************************
*                                                          *
*  Of course, cin. The ALU in this operation is just       *
*  an adder, adding the resulting AND operation with       *
*  zero.                                                   *
*                                                          *
************************************************************
*
lo cin
*
************************************************************
*                                                          *
*  For the ALU, the read and write signals                 *
*  are disabled:                                           *
*                                                          *
************************************************************
*
lo rd1 rd2 wr1
*
************************************************************
*                                                          *
*  The following are for rolls and shifts. They are       *
*  inputs to the select alu output multiplexer. Their      *
*  value in the finished design depends on the carry       *
*  bit in the status register.                             *
*  In this instruction, also not used:                     *
*                                                          *
************************************************************
*
lo slin srin
*
************************************************************
*                                                          *
*  Enabling the alu bus 3-state drivers.                   *
*  By doing this, the ALU result will be allowed to        *
*  appear on the main internal bus. Data for the           *
*  ALU output will be plotted from those nodes.            *
*                                                          *
************************************************************
*
cl busalu 1000ns 11111111111111111
as busalu bsa0 bsa1
*
```

```
*************************************************************
*                                                           *
*  Clocking the register inputs to the ALU:                 *
*                                                           *
*************************************************************
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
```

```
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011110000
as ax14 ax14
cl ax13 1000ns 00000000111111111111
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111111111111111
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011111111
as ax8 ax8
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000001111
as ax7 ax7
cl ax6  1000ns 11111111000011111111
as ax6 ax6
cl ax5  1000ns 00001111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
```

```
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 0000000011111111111
as ax0 ax0
*
**********************************************************
*                                                        *
*  Phew!  Now for the plotting. This is the main bus.    *
*  Also, status bits for the result of the operation.    *
*                                                        *
**********************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
**********************************************************
*                                                        *
*  And, a clock for measuring delays against. This       *
*  clock does not affect the ALU operation.              *
*                                                        *
**********************************************************
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
```

```
*************************************************************
*                                                           *
*  Now, some simulation parameters.                         *
*                                                           *
*  Plot step:                               ps 10ns         *
*  Power Output? ( y = yes ):               po y            *
*  Simulation Length:                       sl 1000ns       *
*                                                           *
*************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.AND
pf ALU.and.out
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:        4.99108 mW                         *
*  Average Current:      0.998217 mA                        *
*                                                           *
*************************************************************
```

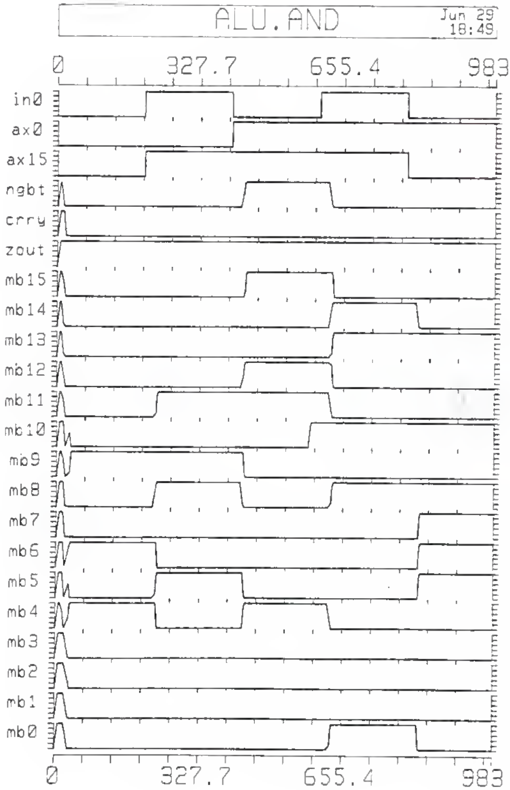

Figure B14: Plot Of Results From ALU.AND Simulation

```
***********************************************************
*                                                         *
*           Simulation file for checking ALU.             *
*                       ALU.OR                            *
*           This is for the OR instruction.               *
*                                                         *
***********************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89.
*
***********************************************************
*                                                         *
*  First, the multiplexing control. The select alu        *
*  control signals are for the 4x1 mux at the output      *
*  of the alu. Used to perform shifts and rotates.        *
*                                                         *
***********************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
lo s01 s02 sb10 sb11
hi s11 s12 sb00 sb01
*
***********************************************************
*                                                         *
*  Control inputs for AX and TMP multiplexing. For        *
*  this instruction, the AX and TMP inputs are ORed       *
*  together from the AND, OR, EXOR multiplexing, then     *
*  added to zero in the adder portion of the ALU.         *
*                                                         *
***********************************************************
*
hi tzro
lo tone aone szro
*
```

```
*************************************************************
*                                                           *
*  Of course, cin. The ALU in this operation is just        *
*  adding w/ no previous carry.                             *
*                                                           *
*************************************************************
*
lo cin
*
*************************************************************
*                                                           *
*  For the ALU, the read and write signals                  *
*  are disabled:                                            *
*                                                           *
*************************************************************
*
lo rd1 rd2 wr1
*
*************************************************************
*                                                           *
*  The following are for rolls and shifts. They are         *
*  inputs to the select alu output multiplexer. Their       *
*  value in the finished design depends on the carry        *
*  bit in the status register.                              *
*  In this instruction, also not used:                      *
*                                                           *
*************************************************************
*
lo slin srin
*
*************************************************************
*                                                           *
*  Enabling the alu bus 3-state drivers.                    *
*  By doing this, the ALU result will be allowed to         *
*  appear on the main internal bus. Data for the            *
*  ALU output will be plotted from those nodes.             *
*                                                           *
*************************************************************
*
cl busalu 1000ns 1111111111111111111
as busalu bsa0 bsa1
*
```

```
**************************************************************
*                                                            *
*  Clocking the register inputs to the ALU:                  *
*                                                            *
**************************************************************
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011110000
as ax14 ax14
cl ax13 1000ns 00000000111111111111
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111111111111111
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011111111
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000001111
as ax7 ax7
cl ax6  1000ns 11111111000011111111
as ax6 ax6
cl ax5  1000ns 00001111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111111111
as ax0 ax0
*
```

```
***********************************************************
*                                                         *
*  Phew!  Now for the plotting. This is the main bus.     *
*  Also, status bits for the result of the operation.     *
*                                                         *
***********************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
***********************************************************
*                                                         *
*  And, a clock for measuring delays against. This        *
*  clock does not affect the ALU operation.               *
*                                                         *
***********************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
***********************************************************
*                                                         *
*  Now, some simulation parameters.                       *
*                                                         *
*  Plot Step:                          ps 10ns            *
*  Power Output? ( y = yes ):          po y               *
*  Simulation length:                  sl 1000ns          *
*                                                         *
***********************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.OR
pf ALU.or.out
*
***********************************************************
*                                                         *
*  Power given after simulation:                          *
*                                                         *
*  Average Power:            4.82982 mW                   *
*  Average Current:          0.96596 mA                   *
*                                                         *
***********************************************************
```

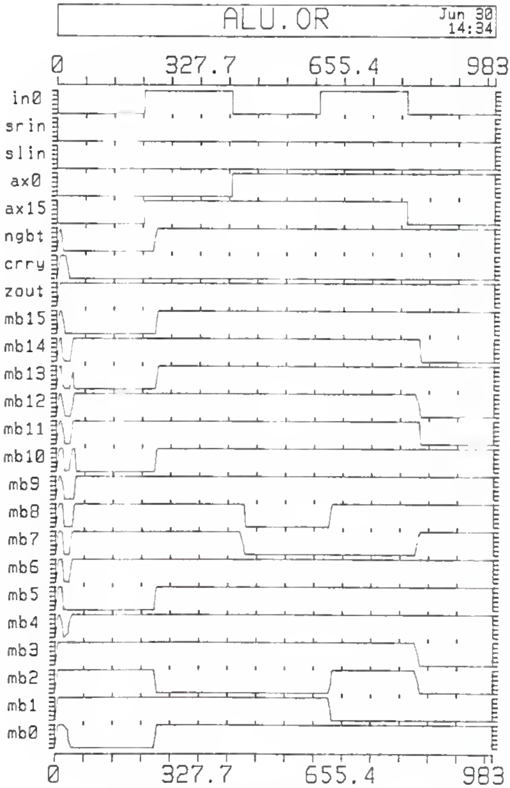

Figure B15: Plot Of Results From ALU.OR Simulation

```
**********************************************************
*                                                        *
*          Simulation file for checking ALU.             *
*                      ALU.XOR                           *
*          This is for the XOR instruction.              *
*                                                        *
**********************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89.
*
**********************************************************
*                                                        *
*  First, the multiplexing control. The select ALU       *
*  control signals are for the 4x1 mux at the output     *
*  of the ALU. Used to perform shifts and rolls.         *
*                                                        *
**********************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
lo s01 s02 s11 s12
hi sb00 sb01 sb10 sb11
*
**********************************************************
*                                                        *
*  Control inputs for AX and TMP multiplexing. For       *
*  this instruction, the AX and TMP inputs are XORed     *
*  together. The OR result is allowed to pass through    *
*  the adder portion of the ALU by adding it to zero.    *
*                                                        *
**********************************************************
*
hi tzro
lo tone aone szro
*
```

```
**************************************************************
*                                                            *
*  Of course, cin. The ALU in this operation is just         *
*  an adder w/ no previous carry.                            *
*                                                            *
**************************************************************
*
lo cin
*
**************************************************************
*                                                            *
*  For the ALU, the read and write signals                   *
*  are disabled:                                             *
*                                                            *
**************************************************************
*
lo rd1 rd2 wr1
*
**************************************************************
*                                                            *
*  The following are for rolls and shifts. They are          *
*  inputs to the select ALU output multiplexer. Their        *
*  value in the finished design depends on the carry         *
*  bit in the status register.                               *
*  In this instruction, also not used:                       *
*                                                            *
**************************************************************
*
lo slin srin
*
**************************************************************
*                                                            *
*  Enabling the alu bus 3-state drivers.                     *
*  By doing this , the ALU result will be allowed to         *
*  appear on the main internal bus. Data for the             *
*  ALU output will be plotted from those nodes.              *
*                                                            *
**************************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsa1
*
```

```
***********************************************************
*                                                         *
*  Clocking the register inputs to the ALU:               *
*                                                         *
***********************************************************
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011110000
as ax14 ax14
cl ax13 1000ns 00000000111111111111
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111111111111111
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011111111
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000001111
as ax7 ax7
cl ax6  1000ns 11111111000011111111
as ax6 ax6
cl ax5  1000ns 00001111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111111111
as ax0 ax0
*
```

```
************************************************************
*                                                          *
*  Phew!  Now for the plotting. This is the main bus.      *
*  Also, status bits for the result of the operation.      *
*                                                          *
************************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
************************************************************
*                                                          *
*  And, a clock for measuring delays against. This         *
*  clock does affect the ALU operation.                    *
*                                                          *
************************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
************************************************************
*                                                          *
*  Now, some simulation parameters:                        *
*                                                          *
*  Plot Step:                          ps 10ns             *
*  Power Output? ( y = yes ):          po y                *
*  Simulation Length:                  sl 1000ns           *
*                                                          *
************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.XOR
pf ALU.xor.out
*
************************************************************
*                                                          *
*  Power given after simulation:                           *
*                                                          *
*  Average Power:           5.36968 mW                     *
*  Average Current:         1.07394 mA                     *
*                                                          *
************************************************************
```

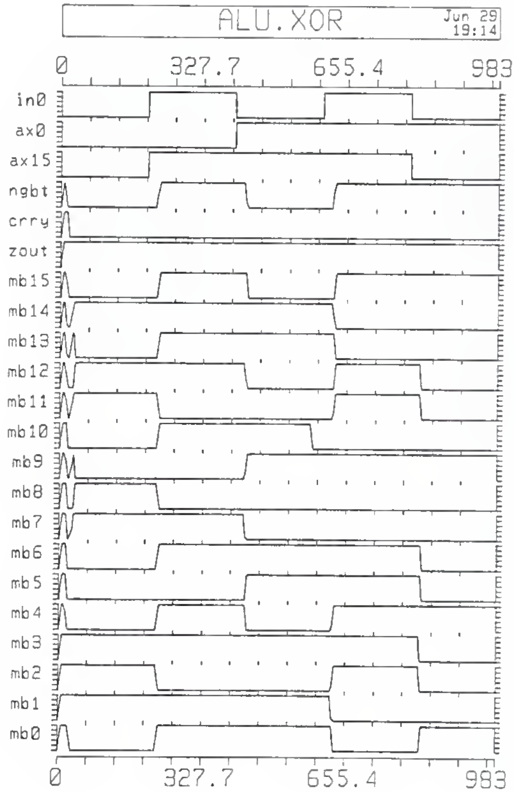

Figure B16: Plot Of Results From ALU.XOR Simulation

```
************************************************************
*                                                          *
*           Simulation file for checking ALU.              *
*                      ALU.CMP                             *
*            This is for the TST instruction.              *
*                                                          *
************************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89
*
************************************************************
*                                                          *
*  First, the multiplexing control. The select ALU         *
*  control signals are for the 4x1 mux at the output of    *
*  the ALU. Used to perform shifts and rolls.              *
*                                                          *
************************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
lo s01 s02 s11 s12
hi sb00 sb01 sb10 sb11
*
************************************************************
*                                                          *
*  Control inputs for AX and TMP multiplexing. For         *
*  this instruction, the AX input is complemented          *
*  using the EXOR multiplexing. The complement is          *
*  then multiplexed through the adder portion of the       *
*  ALU by adding the complement to zero.                   *
*                                                          *
************************************************************
*
lo tone
hi tzro aone szro
*
```

```
***********************************************************
*                                                         *
*  Of course, cin. The ALU in this operation is just      *
*  an added w/ no previous carry.                         *
*                                                         *
***********************************************************
*
hi cin
*
***********************************************************
*                                                         *
*  For the ALU, the read and write signals                *
*  are disabled:                                          *
*                                                         *
***********************************************************
*
lo rd1 rd2 wr1
*
***********************************************************
*                                                         *
*  The following are for rolls and shifts. They are       *
*  inputs to the select ALU output multiplexer . Their    *
*  value in the finished design depends on the carry      *
*  bit in the register.                                   *
*  In this instruction, also not used. They are clocked   *
*  here to show that they have no affect on this          *
*  operation.                                             *
*                                                         *
***********************************************************
*
cl slin 1000ns   00000000111111111111
as slin slin
cl srin 1000ns   11110000111100000000
as srin srin
*
***********************************************************
*                                                         *
*  Enabling the alu bus 3-state drivers.                  *
*  By doing this this, the ALU result will be allowed     *
*  to appear on the main internal bus. Data for the       *
*  ALU output will be plotted from those nodes.           *
*                                                         *
***********************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsa1
*
```

```
**********************************************************
*                                                        *
*  Clocking the register inputs to the ALU:              *
*                                                        *
**********************************************************
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 00001111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 00001111111100000000
as tp12 tp12
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 00001111111100000000
as tp9 tp9
cl tp8  1000ns 00001111000011111111
as tp8 tp8
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 00000000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 00000000111111111111
as tp4 tp4
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 00001111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 00001111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000111111110000
as ax14 ax14
cl ax13 1000ns 00000000000011110000
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111000011110000
as ax10 ax10
cl ax9  1000ns 11110000111111110000
as ax9 ax9
cl ax8  1000ns 00001111000011110000
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000000000
as ax7 ax7
cl ax6  1000ns 11110000111111110000
as ax6 ax6
cl ax5  1000ns 00001111000011110000
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000000011110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00001111000000000000
as ax1 ax1
cl ax0  1000ns 00000000000011110000
as ax0 ax0
*
```

```
***********************************************************
*                                                         *
*  Phew!  Now for the plotting. This is the main bus.     *
*  Also, status bits for the result of the operation.     *
*                                                         *
***********************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0 slin srin
*
***********************************************************
*                                                         *
*  And, a clock for measuring delays against. This        *
*  clock does not affect the ALU operation.               *
*                                                         *
***********************************************************
*
*                |  1  |  2  |  3  |  4  |  5  |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
***********************************************************
*                                                         *
*  Now, some simulation parameters:                       *
*                                                         *
*  Plot Step:                          ps 10ns            *
*  Power Output? ( y = yes ):          po y               *
*  Simulation Length:                  sl 1000ns          *
*                                                         *
***********************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.CMP
pf ALU.cmp.out
*
***********************************************************
*                                                         *
*  Power given after simulation:                          *
*                                                         *
*  Average Power:            6.27019 mW                   *
*  Average Current:          1.25404 mA                   *
*                                                         *
***********************************************************
```

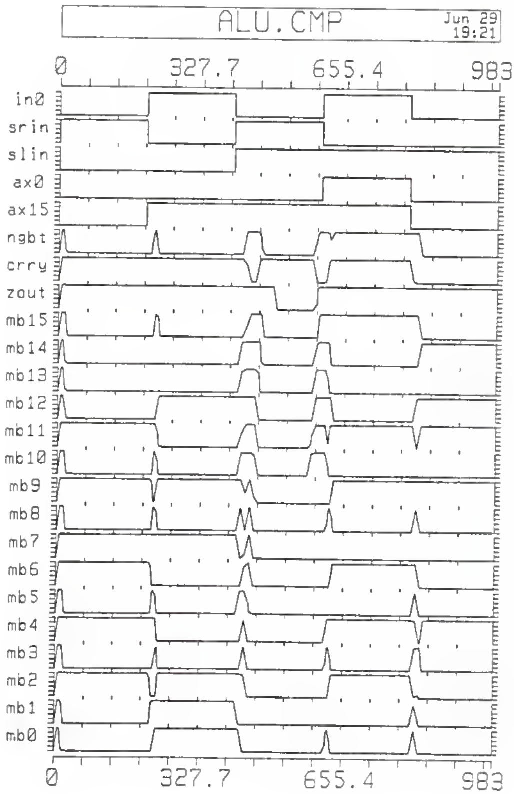

Figure B17: Plot Of Results From ALU.CMP Simulation

```
**************************************************************
*                                                            *
*           Simulation file for checking ALU.                *
*                       ALU.SHR                              *
*            This is for the SHR instruction.                *
*                                                            *
**************************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89
*
**************************************************************
*                                                            *
*  First, the multiplexing control. The select ALU           *
*  control signals are for the 4x1mux at the output of       *
*  the ALU. Used to perform shifts and rolls.                *
*                                                            *
**************************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 xb10 xb11
hi xb00 xb01 x11 x12
*
*  selax(1) & selax(0):
*
lo s01 s02 s11 s12
hi sb00 sb01 sb10 sb11
*
**************************************************************
*                                                            *
*  Control inputs for AX and TMP multiplexing. For this      *
*  instruction, the AX input is "shifted" by using the       *
*  select ALU output multiplexer. The AND, OR, EXOR          *
*  multiplexing and the adder portion of the ALU are         *
*  not used.                                                 *
*                                                            *
**************************************************************
*
lo tzro tone aone szro
lo cin
*
```

```
*************************************************************
*                                                           *
*  For the ALU, the read and write signals                  *
*  are disabled:                                            *
*                                                           *
*************************************************************
*
lo rdl rd2 wrl
*
*************************************************************
*                                                           *
*  The following are for rolls and shifts. They are         *
*  inputs to the select ALU output multiplexer. Their       *
*  value in the finished design depends on the carry        *
*  bit in the status register.                              *
*                                                           *
*************************************************************
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl slin 1000ns   00000000111111111111
as slin slin
cl srin 1000ns   11110000111100000000
as srin srin
*
*************************************************************
*                                                           *
*  Enabling the alu bus 3-state drivers.                    *
*  By doing this, the ALU result will be allowed to         *
*  appear on the main internal bus. Data for the ALU        *
*  output will be plotted from those nodes.                 *
*                                                           *
*************************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsal
*
```

```
*************************************************************
*                                                           *
*  Clocking the register inputs to the ALU:                 *
*                                                           *
*************************************************************
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011110000
as ax14 ax14
cl ax13 1000ns 00000000111111111111
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111111111111111
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011111111
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000001111
as ax7 ax7
cl ax6  1000ns 11111111000011111111
as ax6 ax6
cl ax5  1000ns 00001111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111111111
as ax0 ax0
*
```

```
************************************************************
*                                                          *
*  Phew!  Now for the plotting. This is the main bus.      *
*  Also, status bits for the result of the operation.      *
*                                                          *
************************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0 slin srin
*
************************************************************
*                                                          *
*  And, a clock for measuring delays against. This         *
*  clock does not affect the ALU operation.                *
*                                                          *
************************************************************
*
*                | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
************************************************************
*                                                          *
*  Now, some simulation parameters:                        *
*                                                          *
*  Plot Step:                          ps 10ns             *
*  Power Output? ( y = yes ):          po y                *
*  Simulation Length:                  sl 1000ns           *
*                                                          *
************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.SHR
pf ALU.shr.out
*
************************************************************
*                                                          *
*  Power given after simulation:                           *
*                                                          *
*  Average Power:              3.33506 mW                  *
*  Average Current:            0.667012 mA                 *
*                                                          *
************************************************************
```

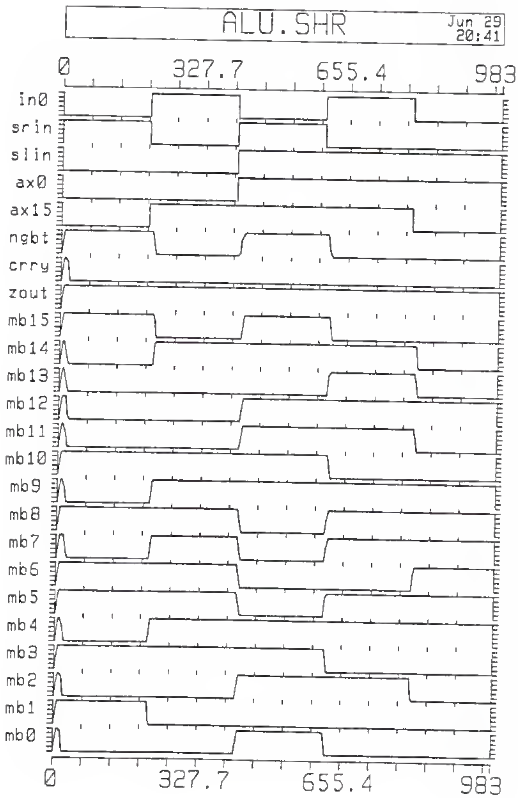

Figure B18: Plot Of Results From ALU.SHR Simulation

```
************************************************************
*                                                          *
*          Simulation file for checking ALU.               *
*                        ALU.SHL                           *
*           This is for the SHL instruction.               *
*                                                          *
************************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89.
*
************************************************************
*                                                          *
*  First, the multiplexing control:                        *
*                                                          *
************************************************************
*
*  selalu(1) & selalu(0):
*
hi x01 x02 x11 x12
lo xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
hi s01 s02 sb10 sb11
lo s11 s12 sb00 sb01
*
************************************************************
*                                                          *
*  Control inputs for AX and TMP and Cin:                  *
*                                                          *
************************************************************
*
hi tzro
lo tone aone szro
lo cin
*
```

```
***********************************************************
*                                                         *
*  For the ALU, the read and write signals                *
*  are disabled:                                          *
*                                                         *
***********************************************************
*
lo rd1 rd2 wr1
*
***********************************************************
*                                                         *
*  The following are for rolls and shifts.                *
*  Here, we shift in a zero.                              *
*                                                         *
***********************************************************
*
lo slin srin
*
***********************************************************
*                                                         *
*  Enabling the alu bus 3-state drivers:                  *
*                                                         *
***********************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsa1
*
***********************************************************
*                                                         *
*  Clocking the register inputs to the ALU:               *
*                                                         *
***********************************************************
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl tpl5 1000ns 00000000111100001111
as tpl5 tpl5
cl tpl4 1000ns 11111111111111110000
as tpl4 tpl4
cl tpl3 1000ns 00001111000011111111
as tpl3 tpl3
cl tpl2 1000ns 11111111111100000000
as tpl2 tpl2
*
```

```
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000111111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
*
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011110000
as ax14 ax14
cl ax13 1000ns 00000000111111111111
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111111111111111
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011111111
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000001111
as ax7 ax7
cl ax6  1000ns 11111111000011111111
as ax6 ax6
cl ax5  1000ns 00001111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111111111
as ax0 ax0
*
*************************************************************
*                                                           *
*  Phew!  Now for the plotting:                             *
*                                                           *
*************************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
```

```
*************************************************************
*                                                           *
*  And, a clock for measuring delays                        *
*  against:                                                 *
*                                                           *
*************************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
*************************************************************
*                                                           *
*  Now, some simulation parameters:                         *
*                                                           *
*  Plot Step:                        ps 10ns                *
*  Power Output? ( y = yes ):        po y                   *
*  Simulation Length:                sl 1000ns              *
*                                                           *
*************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.SHL
pf ALU.shl.out
*
*************************************************************
*                                                           *
*  Power given from FACTS after simulation:                 *
*                                                           *
*  Average Power:              4.96592  milliwatts          *
*  Average Current:            0.99318  milliamps           *
*                                                           *
*************************************************************
```

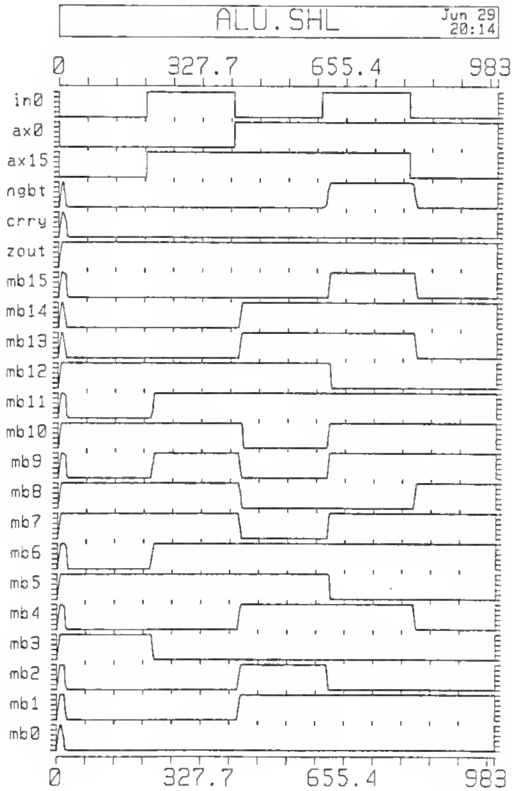

Figure B19: Plot Of Result From ALU.SHL Simulation

```
************************************************************
*                                                          *
*           Simulation file for checking ALU.             *
*                        ALU.INC                           *
*            This is for the INC instruction.              *
*                                                          *
************************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated 2/19/89 after MAJOR
*  redesign due to subtraction
*  operation error.
*
************************************************************
*                                                          *
*  First, the multiplexing control:                        *
*                                                          *
************************************************************
*
*  selalu(1) & selalu(0):
*
hi xb00 xb01 xb10 xb11
lo x01 x02 x11 x12
*
*  selax(1) & selax(0):
*
lo s11 s12 sb00 sb01
hi s01 s02 sb10 sb11
*
************************************************************
*                                                          *
*  Control inputs for AX and TMP and Cin:                  *
*                                                          *
************************************************************
*
hi szro
lo tzro tone aone
hi cin
*
```

```
************************************************************
*                                                          *
*  For the ALU, the read and write signals                 *
*  are disabled:                                           *
*                                                          *
************************************************************
*
lo rd1 rd2 wr1
*
************************************************************
*                                                          *
*  The following are for rolls and shifts.                 *
*  In this instruction, also not used:                     *
*                                                          *
************************************************************
*
lo slin srin
*
************************************************************
*                                                          *
*  Enabling the alu bus 3-state drivers:                   *
*                                                          *
************************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsa1
*
************************************************************
*                                                          *
*  Clocking the register inputs to the ALU:                *
*                                                          *
************************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000111111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 11110000111100001111
as ax15 ax15
cl ax14 1000ns 11111111000011111111
as ax14 ax14
cl ax13 1000ns 11110000000011110000
as ax13 ax13
cl ax12 1000ns 11110000000011111111
as ax12 ax12
*
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111000011111111
as ax11 ax11
cl ax10 1000ns 11111111000000001111
as ax10 ax10
cl ax9  1000ns 11111111111111110000
as ax9 ax9
cl ax8  1000ns 11111111000000000000
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111111100001111
as ax7 ax7
cl ax6  1000ns 11111111111111110000
as ax6 ax6
cl ax5  1000ns 11111111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 11111111111111111111
as ax3 ax3
cl ax2  1000ns 11111111111111111111
as ax2 ax2
cl ax1  1000ns 11111111111111111111
as ax1 ax1
cl ax0  1000ns 11111111111111111111
as ax0 ax0
*
********************************************************
*                                                      *
*  Phew!  Now for the plotting:                        *
*                                                      *
********************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
```

```
*************************************************************
*                                                           *
*  And, a clock for measuring delays                        *
*  against:                                                 *
*                                                           *
*************************************************************
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
*************************************************************
*                                                           *
*  Now, some simulation parameters:                         *
*                                                           *
*  Plot Step:                         ps 10ns               *
*  Power Output? ( y = yes ):         po y                  *
*  Simulation Length:                 sl 1000ns             *
*                                                           *
*************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.INC
pf ALU.inc.out
*
*************************************************************
*                                                           *
*  Power given from FACTS after simulation:                 *
*                                                           *
*  Average Power:          5.69635  milliwatts              *
*  Average Current:        1.13927  milliamps               *
*                                                           *
*************************************************************
```

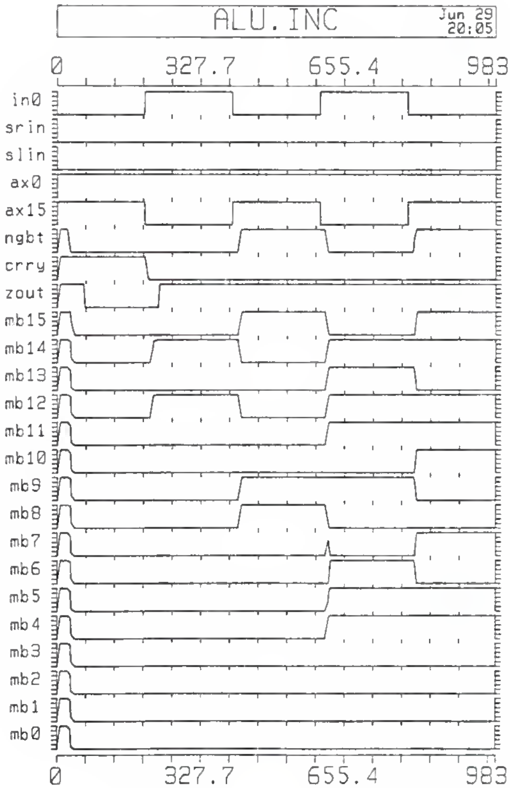

Figure B20: Plot Of Results From ALU.INC Simulation

```
*************************************************************
*                                                           *
*           Simulation file for checking ALU.               *
*                       ALU.DEC                             *
*            This is for the DEC instruction.               *
*                                                           *
*************************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated 2/19/89 after MAJOR
*  redesign due to subtraction
*  operation error.
*
*************************************************************
*                                                           *
*  First, the multiplexing control. The select alu          *
*  control signals are for the 4x1 mux at the output of     *
*  the alu. Used to perform shifts and rolls.               *
*                                                           *
*************************************************************
*
*  selalu(1) & selalu(0):
*
hi xb00 xb01 xb10 xb11
lo x01 x02 x11 x12
*
*  selax(1) & selax(0):
*
hi s11 s12 sb00 sb01
lo s01 s02 sb10 sb11
*
*************************************************************
*                                                           *
*  Control inputs for AX and TMP multiplexing. For          *
*  this instruction, the AX input is decremented.           *
*  This is accomplished by forcing one input of the         *
*  adder portion of the alu to FFFF and adding it to        *
*  AX.                                                      *
*                                                           *
*************************************************************
*
hi szro tone
lo tzro aone
*
```

```
************************************************************
*                                                          *
*  Of course, cin. Since this instruction performs a       *
*  subtraction, cin is set high to accomplish unsigned     *
*  2's complement subtraction.                             *
*                                                          *
************************************************************
lo cin
*
************************************************************
*                                                          *
*  For the ALU, the read and write signals                 *
*  are disabled:                                           *
*                                                          *
************************************************************
*
lo rdl rd2 wrl
*
************************************************************
*                                                          *
*  The following are for rolls and shifts. They are       *
*  inputs to the select alu output multiplexer. Their      *
*  value in the finished design depends on the carry       *
*  bit in the status register.                             *
*  In this instruction, also not used:                     *
*                                                          *
************************************************************
*
lo slin srin
*
************************************************************
*                                                          *
*  Enabling the alu bus 3-state drivers.                   *
*  By doing this, the ALU result will allowed to appear    *
*  on the main internal bus. Data for the ALU output       *
*  will be plotted from those nodes.                       *
*                                                          *
************************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl busalu 1000ns 1111111111111111111
as busalu bsa0 bsal
*
```

```
**********************************************************
*                                                        *
*  Clocking the register inputs to the ALU:              *
*                                                        *
**********************************************************
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                    | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00001111111111110000
as ax14 ax14
cl ax13 1000ns 00001111000011110000
as ax13 ax13
cl ax12 1000ns 00001111111111110000
as ax12 ax12
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111000011110000
as ax10 ax10
cl ax9  1000ns 11111111000011110000
as ax9 ax9
cl ax8  1000ns 00001111000011110000
as ax8 ax8
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000000000
as ax7 ax7
cl ax6  1000ns 11111111111111110000
as ax6 ax6
cl ax5  1000ns 00001111000011110000
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                   | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000n  00001111000011110000
as ax3 ax3
cl ax2  1000ns 11111111000000000000
as ax2 ax2
cl ax1  1000ns 00001111000000000000
as ax1 ax1
cl ax0  1000ns 00001111000011110000
as ax0 ax0
*
```

```
***********************************************************
*                                                         *
*  Phew!  Now for the plotting. This is the main bus.     *
*  Also, status bits for the result of the operation.     *
*                                                         *
***********************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
***********************************************************
*                                                         *
*  And, a clock for measuring delays against. This        *
*  clock affect the ALU operation.                        *
*                                                         *
***********************************************************
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
***********************************************************
*                                                         *
*  Now, some simulation parameters.                       *
*                                                         *
*  Plot Step:                        ps 10ns              *
*  Power Output? ( y = yes ):        po y                 *
*  Simulation Length:                sl 1000ns            *
*                                                         *
***********************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.DEC
pf ALU.dec.out
*
***********************************************************
*                                                         *
*  Power given after simulation:                          *
*                                                         *
*  Average Power:      5.66469  milliwatts                *
*  Average Current:    1.13294  milliamps                 *
*                                                         *
***********************************************************
```

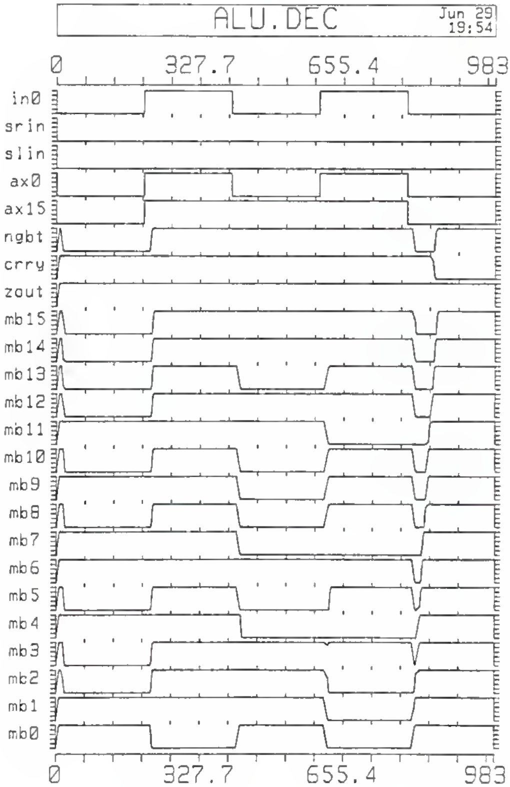

Figure B21: Plot Of Results From ALU.DEC Simulation

```
**********************************************************
*                                                        *
*            Simulation file for checking ALU.           *
*                       ALU.COM                          *
*             This is for the COM instruction.           *
*                                                        *
**********************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89
*
**********************************************************
*                                                        *
*  First, the multiplexing control. The select ALU       *
*  control signals are for the 4x1 mux at the output     *
*  of the ALU. Used to perform shifts and rolls.         *
*                                                        *
**********************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
lo s01 s02 s11 s12
hi sb00 sb01 sb10 sb11
*
**********************************************************
*                                                        *
*  Control inputs for AX and TMP multiplexing. For this  *
*  instruction, the EXOR multiplexing is used to         *
*  complement the AX input. The result is added to       *
*  zero in the adder portion of the ALU.                 *
*                                                        *
**********************************************************
*
lo tzro aone szro
hi tone
lo cin
*
```

```
***********************************************************
*                                                         *
*  For the ALU, the read and write signals                *
*  are disabled:                                          *
*                                                         *
***********************************************************
*
lo rdl rd2 wrl
*
***********************************************************
*                                                         *
*  The following are for rolls and shifts. They are       *
*  inputs to the select ALU output multiplexer. Their     *
*  value in the finished design depends on the carry      *
*  bit in the status register.                            *
*  For this instruction, also not used.                   *
*                                                         *
***********************************************************
*
cl slin 1000ns   00000000111111111111
as slin slin
cl srin 1000ns   11110000111100000000
as srin srin
*
***********************************************************
*                                                         *
*  Enabling the alu bus 3-state drivers.                  *
*  By doing this, the ALU result will be allowed to       *
*  appear on the main internal bus. Data for the          *
*  ALU output will be plotted from those nodes.           *
*                                                         *
***********************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsal
*
```

```
*************************************************************
*                                                           *
*  Clocking the register inputs to the ALU:                 *
*                                                           *
*************************************************************
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011110000
as ax14 ax14
cl ax13 1000ns 00000000111111111111
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111111111111111
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011111111
as ax8 ax8
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000001111
as ax7 ax7
cl ax6  1000ns 11111111000011111111
as ax6 ax6
cl ax5  1000ns 00001111111111111111
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111111111
as ax0 ax0
*
```

```
************************************************************
*                                                          *
*  Phew!  Now for the plotting. This is the main bus.      *
*  Also, status bits for the result of the operation.      *
*                                                          *
************************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0 slin srin
*
************************************************************
*                                                          *
*  And, a clock for measuring delays against. This         *
*  clock does not affect the ALU operation.                *
*                                                          *
************************************************************
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0*
************************************************************
*                                                          *
*  Now, some simulation parameters:                        *
*                                                          *
*  Plot Step:                          ps 10ns             *
*  Power Output? ( y = yes ):          po y                *
*  Simulation Length:                  sl 1000ns           *
*                                                          *
************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.COM
pf ALU.com.out
*
************************************************************
*                                                          *
*  Power given from simulation:                            *
*                                                          *
*  Average Power:          3.31195 mW                      *
*  Average Current:        0.66239 mA                      *
*                                                          *
************************************************************
```

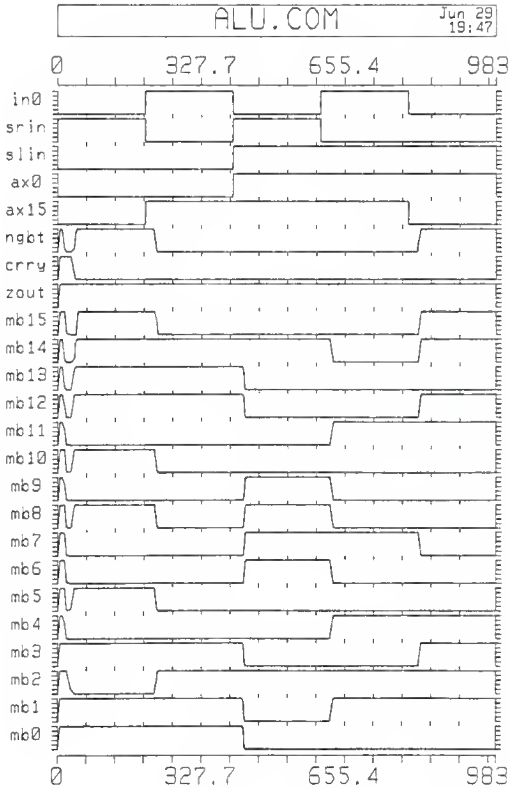

Figure B22: Plot Of Results From ALU.COM Simulation

```
*************************************************************
*                                                           *
*           Simulation file for checking ALU.               *
*                      ALU.TST                              *
*            This is for the TST instruction.               *
*                                                           *
*************************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89
*
*************************************************************
*                                                           *
*  First, the multiplexing control. The select ALU          *
*  control signals are for the 4x1 mux at the output        *
*  of the ALU. Used to perform shifts and rolls.            *
*                                                           *
*************************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
lo s01 s02 s11 s12
hi sb00 sb01 sb10 sb11
*
*************************************************************
*                                                           *
*  Control inputs for AX and TMP multiplexing. For this     *
*  instruction, the AX input complemented using the         *
*  EXOR multiplexing and added to zero. The CIN input       *
*  is sent high.                                            *
*                                                           *
*************************************************************
*
lo tzro tone
hi aone szro
hi cin
*
```

```
***********************************************************
*                                                         *
*  For the ALU, the read and write signals                *
*  are disabled:                                          *
*                                                         *
***********************************************************
*
lo rdl rd2 wrl
*
***********************************************************
*                                                         *
*  The following are for rolls and shifts. They are       *
*  inputs to the select ALU output multiplexer. Their     *
*  value in the finished design depends on the carry      *
*  bit in the status register.                            *
*  In this instruction, also not used.                    *
*                                                         *
***********************************************************
*
cl slin 1000ns   00000000111111111111
as slin slin
cl srin 1000ns   11110000111100000000
as srin srin
*
***********************************************************
*                                                         *
*  Enabling the alu bus 3-state drivers.                  *
*  By doing this, the ALU result will be allowed to       *
*  appear on the main internal bus. Data for the          *
*  ALU output will be plotted from those nodes.           *
*                                                         *
***********************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsal
*
```

```
**************************************************************
*                                                            *
*  Clocking the register inputs to the ALU:                  *
*
**************************************************************
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 00001111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 00001111111100000000
as tp12 tp12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 00001111111100000000
as tp9 tp9
cl tp8  1000ns 00001111000011111111
as tp8 tp8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 00000000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 00000000111111111111
as tp4 tp4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 00001111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 00001111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000111111110000
as ax14 ax14
cl ax13 1000ns 00000000000011110000
as ax13 ax13
cl ax12 1000ns 00000000111111110000
as ax12 ax12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100000000
as ax11 ax11
cl ax10 1000ns 00001111000011110000
as ax10 ax10
cl ax9  1000ns 11110000111111110000
as ax9 ax9
cl ax8  1000ns 00001111000011110000
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000000000
as ax7 ax7
cl ax6  1000ns 11110000111111110000
as ax6 ax6
cl ax5  1000ns 00001111000011110000
as ax5 ax5
cl ax4  1000ns 11111111111100000000
as ax4 ax4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000000011110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00001111000000000000
as ax1 ax1
cl ax0  1000ns 00000000000011110000
as ax0 ax0
*
```

```
************************************************************
*                                                          *
*  Phew!  Now for the plotting. This is the main bus.      *
*  Also, status bits for the result of the operation.      *
*                                                          *
************************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0 slin srin
*
************************************************************
*                                                          *
*  And, a clock for measuring delays against. This         *
*  clock does not affect the ALU operation.                *
*                                                          *
************************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
************************************************************
*                                                          *
*  Now, some simulation parameters:                        *
*                                                          *
*  Plot Step:                          ps 10ns             *
*  Power Output? ( y = yes ):          po y                *
*  Simulation Length:                  sl 1000ns           *
*                                                          *
************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.TST
pf ALU.tst.out
*
************************************************************
*                                                          *
*  Power given after simulation:                           *
*                                                          *
*  Average Power:        2.79122 mW                        *
*  Average Current:      0.558244 mA                       *
*                                                          *
************************************************************
```

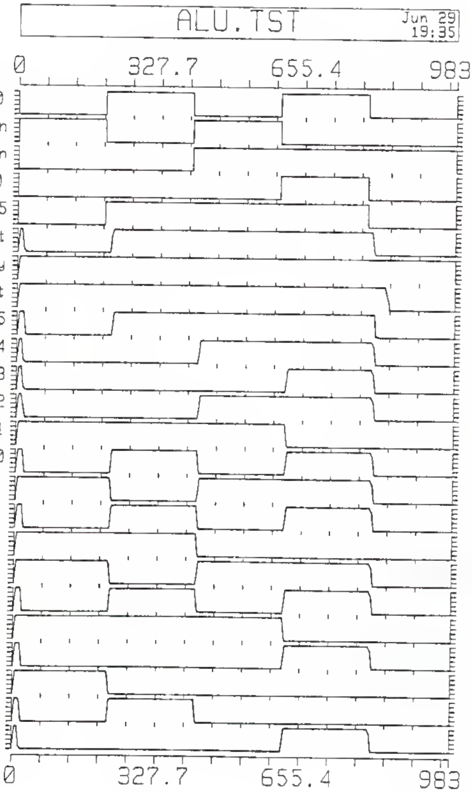

Figure B23: Plot Of Results From ALU.TST Simulation

```
**********************************************************
*                                                        *
*           Simulation file for checking ALU.            *
*                       ALU.BUS                          *
*               This is for checking proper              *
*          operation of the three-state buffers.         *
*                                                        *
**********************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89.
*
**********************************************************
*                                                        *
*  This file will generate an ADD of the AX and TMP      *
*  inputs.                                               *
*                                                        *
*  First, the multiplexing control. The select ALU       *
*  control signals are for the 4x1 mux at the output     *
*  of the ALU. Used to perform shifts and rolls.         *
*                                                        *
**********************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
hi s01 s02 sb10 sb11
lo s11 s12 sb00 sb01
*
```

```
***********************************************************
*                                                         *
*  Control inputs for AX and TMP multiplexing. For        *
*  this instruction, the AX and TMP inputs are added      *
*  together. The TMP is allowed to pass through the       *
*  AND, OR, and EXOR multiplexing, and is then added      *
*  to the AX by the adder portion of the ALU.             *
*                                                         *
***********************************************************
*
hi tzro aone szro
lo tone
*
***********************************************************
*                                                         *
*  Of course, cin. The ALU in this operation is just      *
*  an adder w/ no previous carry.                         *
*                                                         *
***********************************************************
*
lo cin
*
***********************************************************
*                                                         *
*  Here, the read and write clocks are provided to        *
*  enable and disable the trim-state cells at the
*
*  output of the ALU.                                     *
*                                                         *
***********************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl read  1000ns 00001111000011110000
as read rd1 rd2
*
***********************************************************
*                                                         *
*  The following are for rolls and shifts. They are      *
*  inputs to the select ALU output multiplexer. Their     *
*  value in the finished design depends on the carry      *
*  bit in the status register.                            *
*  In this instruction, also not used.                    *
*                                                         *
***********************************************************
*
lo slin srin
*
```

```
***********************************************************
*                                                         *
*  Enabling the alu bus 3-state drivers.                  *
*  By doing this, the ALU result will be allowed to       *
*  appear on the main internal bus. Data for the ALU      *
*  output will be plotted from those nodes.               *
*                                                         *
***********************************************************
*
*
*                | 1 | 2 | 3 | 4 | 5 |
*
cl busalu 1000ns 11110000111100001111
as busalu bsa0 bsa1
*
***********************************************************
*                                                         *
*  Clocking the register inputs to the ALU:               *
*                                                         *
***********************************************************
*
*                | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*                | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
*
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011111111
as ax14 ax14
cl ax13 1000ns 00000000111111110000
as ax13 ax13
cl ax12 1000ns 00000000111111111111
as ax12 ax12
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100001111
as ax11 ax11
cl ax10 1000ns 00001111111111110000
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011110000
as ax8 ax8
*
```

```
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000000000
as ax7 ax7
cl ax6  1000ns 11111111000011110000
as ax6 ax6
cl ax5  1000ns 00001111111111110000
as ax5 ax5
cl ax4  1000ns 11111111111100001111
as ax4 ax4
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111110000
as ax0 ax0
*
**********************************************************
*                                                        *
*  Creating a clock for the INPUT bus. This data is the  *
*  data read into the TORO during load, indexed, and     *
*  other ALU instructions.                               *
*                                                        *
**********************************************************
*
*                  | 1 | 2 | 3 | 4 | 5 |
*
cl in15 100ns  00001111111100001111
as in15 in15 in11 in7 in3
cl in14 1000ns 00000000000011111111
as in14 in14 in10 in6 in2
cl in13 1000ns 00001111111100000000
as in13 in13 in9 in5 in1
cl in12 1000ns 11110000111100001111
as in12 in12 in8 in4 in0
*
```

```
*************************************************************
*                                                           *
*  Phew!  Now for the plotting. This is the main bus.       *
*  Also, status bits for the result of the operation.       *
*                                                           *
*  ( Note: This simulation file was used before the         *
*    addition of the write register. The output OUT         *
*    is now the output of the write register. Its           *
*    value will not change if this file is run on the       *
*    updated ALU design. )                                  *
*                                                           *
*************************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl out0 out1 out2 out3 out4 out5 out6 out7
pl out8 out9 out10 out11 out12 out13 out14 out15
*
*************************************************************
*                                                           *
*  Now, some simulation parameters:                         *
*                                                           *
*  Plot Step:                          ps 10ns              *
*  Power Output? ( y = yes ):          po y                 *
*  Simulation Length:                  sl 1000ns            *
*                                                           *
*************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.BUS
pf ALU.bus.out
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:          10.3635 mW                       *
*  Average Current:         2.07269 mA                      *
*                                                           *
*************************************************************
```

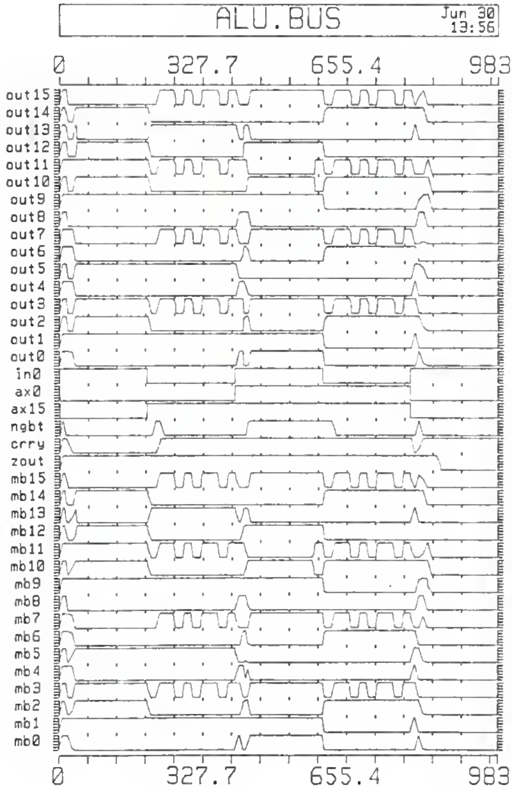

Figure B24: Plot Of Results From ALU.BUS Simulation

```
***********************************************************
*                                                         *
*            Simulation file for checking ALU.            *
*                        ALU.ADD                          *
*            This is for the ADD instruction.             *
*                                                         *
***********************************************************
*
*  Updated 2/16/89 after circuit
*  verification and modification
*  of zero and carry circuits.
*
*  Updated after discovery of MAJOR
*  error in subtraction operations.
*  2/18/89.
*
***********************************************************
*                                                         *
*  First, the multiplexing control: the select alu        *
*  control signals are for the 4x1 mux at the output of   *
*  the alu. Used to perform shifts and rolls.             *
*                                                         *
***********************************************************
*
*  selalu(1) & selalu(0):
*
lo x01 x02 x11 x12
hi xb00 xb01 xb10 xb11
*
*  selax(1) & selax(0):
*
hi s01 s02 sb10 sb11
lo s11 s12 sb00 sb01a
*
***********************************************************
*                                                         *
*  Control inputs for AX and TMP multiplexing. For        *
*  instruction, the AX and TMP inputs are added           *
*  together. The TMP is allowed to pass through the       *
*  AND, OR, and EXOR multiplexing, and is then added      *
*  to the AX by the adder portion of the ALU.             *
*                                                         *
***********************************************************
*
hi tzro aone szro
lo tone
*
```

```
**************************************************************
*                                                            *
*  Of course, cin. The ALU in this operation is just         *
*  an adder w/ no previous carry.                            *
*                                                            *
**************************************************************
*
lo cin
*
*************************************************************
*                                                           *
*  For the ALU, the read and write signals                  *
*  are disabled:                                            *
*                                                           *
*************************************************************
*
lo rd1 rd2 wr1
*
*************************************************************
*                                                           *
*  The following are for rolls and shifts. They are         *
*  inputs to the select alu output multiplexer. Their       *
*  value in the finished design depends on the carry        *
*  bit in the status register.                              *
*  In this instruction, also not used:                      *
*                                                           *
*************************************************************
*
lo slin srin
*
*************************************************************
*                                                           *
*  Enabling the alu bus 3-state drivers.                    *
*  By doing this, the ALU result will be allowed to         *
*  appear on the main internal bus. Data for the            *
*  ALU output will be plotted from those nodes.             *
*                                                           *
*************************************************************
*
cl busalu 1000ns 11111111111111111111
as busalu bsa0 bsa1
*
```

```
*********************************************************
*                                                       *
*  Clocking the register inputs to the ALU:             *
*                                                       *
*********************************************************
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl tp15 1000ns 00000000111100001111
as tp15 tp15
cl tp14 1000ns 11111111111111110000
as tp14 tp14
cl tp13 1000ns 00001111000011111111
as tp13 tp13
cl tp12 1000ns 11111111111100000000
as tp12 tp12
*
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl tp11 1000ns 00001111111111110000
as tp11 tp11
cl tp10 1000ns 00000000000011111111
as tp10 tp10
cl tp9  1000ns 11111111111100000000
as tp9 tp9
cl tp8  1000ns 11111111000011111111
as tp8 tp8
*
*               | 1 | 2 | 3 | 4 | 5 |
*
cl tp7  1000ns 00000000000000001111
as tp7 tp7
cl tp6  1000ns 11110000111100001111
as tp6 tp6
cl tp5  1000ns 00001111000000001111
as tp5 tp5
cl tp4  1000ns 11110000111111111111
as tp4 tp4
*
```

```
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl tp3  1000ns 11111111000000000000
as tp3 tp3
cl tp2  1000ns 00000000000011110000
as tp2 tp2
cl tp1  1000ns 11111111000000000000
as tp1 tp1
cl tp0  1000ns 00001111000011110000
as tp0 tp0
*
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax15 1000ns 00001111111111110000
as ax15 ax15
cl ax14 1000ns 00000000000011111111
as ax14 ax14
cl ax13 1000ns 00000000111111110000
as ax13 ax13
cl ax12 1000ns 00000000111111111111
as ax12 ax12
*
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax11 1000ns 11111111111100001111
as ax11 ax11
cl ax10 1000ns 00001111111111110000
as ax10 ax10
cl ax9  1000ns 11111111000011111111
as ax9 ax9
cl ax8  1000ns 00001111000011110000
as ax8 ax8
*
*                 | 1 | 2 | 3 | 4 | 5 |
*
cl ax7  1000ns 11111111000000000000
as ax7 ax7
cl ax6  1000ns 11111111000011110000
as ax6 ax6
cl ax5  1000ns 00001111111111110000
as ax5 ax5
cl ax4  1000ns 11111111111100001111
as ax4 ax4
*
```

```
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl ax3  1000ns 00000000111111110000
as ax3 ax3
cl ax2  1000ns 11110000000000000000
as ax2 ax2
cl ax1  1000ns 00000000111100000000
as ax1 ax1
cl ax0  1000ns 00000000111111110000
as ax0 ax0
*
**********************************************************
*                                                        *
*  Phew!  Now for the plotting. This is the main bus.    *
*  Also, status bits for the result of the operation.    *
*                                                        *
**********************************************************
*
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
pl zout crry ngbt ax15 ax0
*
**********************************************************
*                                                        *
*  And, a clock for measuring delays against. This       *
*  clock does not affect the ALU operation.              *
*                                                        *
**********************************************************
*
*              | 1 | 2 | 3 | 4 | 5 |
*
cl clock 1000ns 00001111000011110000
as clock in0
pl in0
*
```

```
*************************************************************
*                                                           *
*  Now, some simulation parameters.                         *
*                                                           *
*  Plot step:                             ps 10ns           *
*  Power Output? ( y = yes ):             po y              *
*  Simulation Length:                     sl 1000ns         *
*                                                           *
*************************************************************
*
ps 10ns
po y
sl 1000ns
cm + hpr
ti ALU.ADD
pf ALU.add.out
*
*************************************************************
*                                                           *
*  Power Given after Simulation:                            *
*                                                           *
*  Average Power:          8.093837 mW                      *
*  Average Current:        1.61967 mA                       *
*                                                           *
*************************************************************
```

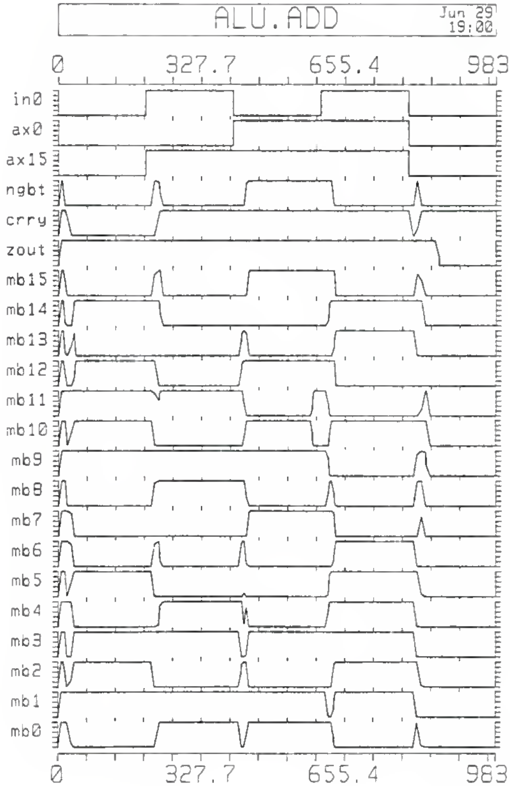

Figure B25: Plot Of Results From ALU.ADD Simulation

8.4 Control Logic Library

Nineteen simulation were performed on the control logic for the TORO machine. There were four sets of tests: one for control signal generation for the addressing and instruction class decode circuitry, one for ALU control signals, one for the branch control signal decode circuitry, and one for the status register. For the last two sets of tests, the tests were not exhaustive. The large number of transistors in the final control logic design and the number of possible input combinations made this task impossible. Tests were performed on what was considered the most used branch instructions and carry bit configurations. Below are descriptions for each simulation.

The following sixteen tests were performed to test the control logic for proper TORO control signal generation. Here, it was a simple matter of applying the proper inputs to the control logic for the instruction class and addressing mode, and supplying a clock. The simulations show that the proper sequence of control signals were generated.

LOAD Indexed: This simulation checked for correct LOAD instruction control signal generation for indexed addressing.

LOAD Direct: This simulation checked for correct LOAD instruction control signal generation for direct addressing.

LOAD Immediate: This simulation checked for correct LOAD instruction control signal generation for immediate addressing.

STORE Indexed: This simulation checked for correct STORE instruction control signal generation for indexed addressing.

STORE Direct: This simulation checked for correct STORE instruction control signal generation for direct addressing.

ALU Indexed: This simulation checked for correct ALU instruction control signal generation for the TORO for indexed addressing.

ALU Direct: This simulation checked for correct ALU instruction control signal generation for the TORO for direct addressing.

ALU Immediate: This simulation checked for correct ALU instruction control signal generation for the TORO for immediate addressing.

ALU Inherent: This simulation checked for correct ALU instruction control signal generation for the TORO for inherent addressing.

```
**********************************************************
*                                                        *
*          Simulation File for checking TORO             *
*            load instruction control signal             *
*        generation during indexed addressing.           *
*                      CNTL.SIM1                         *
*                                                        *
**********************************************************
*
**********************************************************
*                                                        *
*  Setting the instruction class and addressing mode     *
*  inputs to load and indexed. Setting the clock to      *
*  5 MHz.                                                *
*                                                        *
**********************************************************
*
hi m1 m0
lo c0 c1
cl clock 200ns 01
as clock clock
*
**********************************************************
*                                                        *
*  These are the names of the control                    *
*  signals that drive the control signal                 *
*  buffers:                                              *
*                                                        *
*  pl ldpc ldmr T1 ldr ldst ldtp clrt bspc busr          *
*  pl balu read index rgsel denb adenb                   *
*                                                        *
*  and the following are the corresponding               *
*  outputs driver signal names:                          *
*                                                        *
**********************************************************
*
pl ldpcr ldmar ldinr ldr1 ldr2 ldtmp bspcr bsr1 bsr2
pl bsa1 bsa2 rd1 rd2 wr1 rgs1b rgs11 rgs12 rgs13
*
**********************************************************
*                                                        *
*  These signals are for driving the pad                 *
*  enables and the R/W signal to the peripherals:        *
*                                                        *
**********************************************************
*
pl denb adenb rw
*
```

```
*************************************************************
*                                                           *
*  These signals are controls, but internal                 *
*  to the CNTL functional block:                            *
*                                                           *
*************************************************************
*
pl ldst clrt index
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                        ps 10ns                *
*  Power Output? ( y = yes ):        po y                   *
*  Simulation Length:                sl 1200ns              *
*                                                           *
*************************************************************
*
cl reset 1200ns 0111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf ldins.ClMx
ti LOAD Indexed
po y
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:    Not Recorded                           *
*  Average Current:  Not Recorded                           *
*                                                           *
*************************************************************
```

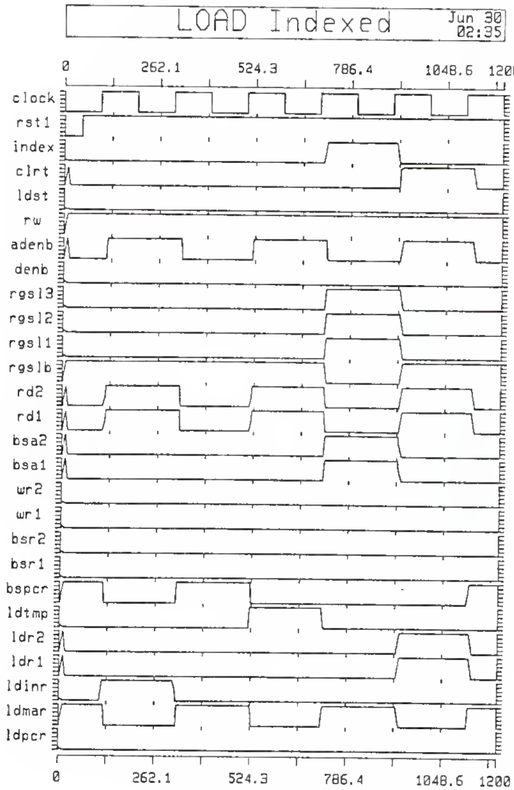

Figure B26: Plot of Results From Load Indexed Instruction Simulation.

```
***********************************************************
*                                                         *
*           Simulation File for checking TORO             *
*            load instruction control signal              *
*          generation during direct addressing.           *
*                        CNTL.SIM2                        *
*                                                         *
***********************************************************
*
***********************************************************
*                                                         *
*  Setting the instruction class and addressing mode      *
*  inputs to load and indexed. Setting the clock to       *
*  5 MHz.                                                 *
*                                                         *
***********************************************************
*
hi ml
lo c0 cl m0
cl clock 200ns 01
as clock clock
*
***********************************************************
*                                                         *
*  These are the names of the control                     *
*  signals that drive the control signal                  *
*  buffers:                                               *
*                                                         *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr           *
*  pl balu read index rgsel denb adenb                    *
*                                                         *
*  and the following are the corresponding                *
*  outputs driver signal names:                           *
*                                                         *
***********************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
***********************************************************
*                                                         *
*  These signals are for driving the pad                  *
*  enables and the R/W signal to the peripherals:         *
*                                                         *
***********************************************************
*
pl denb adenb rw
*
```

```
*************************************************************
*                                                           *
*  These signals are controls, but internal                 *
*  to the CNTL functional block:                            *
*                                                           *
*************************************************************
*
pl ldst clrt index
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                       ps 10ns                 *
*  Power Output? ( y = yes ):       po y                    *
*  Simulation Length:               sl 1200ns               *
*                                                           *
*************************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf ldins.ClMd
ti LOAD Direct
po y
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:       Not Recorded                        *
*  Average Current:     Not Recorded                        *
*                                                           *
*************************************************************
```

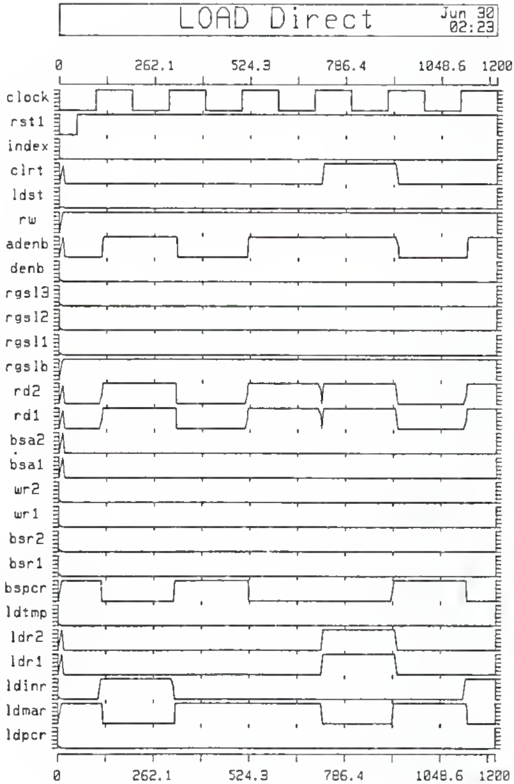

Figure B27: Plot Of Results From Load Direct Instruction Simulation

```
************************************************************
*                                                          *
*           Simulation File for checking TORO              *
*            load instruction control signal               *
*          generation during immediate addressing.         *
*                       CNTL.SIM3                          *
*                                                          *
************************************************************
*
************************************************************
*                                                          *
*  Setting the instruction class and addressing mode       *
*  inputs to load and indexed. Setting the clock to        *
*  5 MHz.                                                  *
*                                                          *
************************************************************
*
hi m0
lo c0 cl ml
cl clock 200ns 01
as clock clock
*
************************************************************
*                                                          *
*  These are the names of the control                      *
*  signals that drive the control signal                   *
*  buffers:                                                *
*                                                          *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr            *
*  pl balu read index rgsel denb adenb                     *
*                                                          *
*  and the following are the corresponding                 *
*  outputs driver signal names:                            *
*                                                          *
************************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsl1 rgsl2 rgsl3
*
************************************************************
*                                                          *
*  These signals are for driving the pad                   *
*  enables and the R/W signal to the peripherals:          *
*                                                          *
************************************************************
*
pl denb adenb rw
*
```

```
***********************************************************
*                                                         *
*  These signals are controls, but internal               *
*  to the CNTL functional block:                          *
*                                                         *
***********************************************************
*
pl ldst clrt index
*
***********************************************************
*                                                         *
*  Simulation Parameters:                                 *
*                                                         *
*  Plot Step:                       ps 10ns               *
*  Power Output? ( y = yes ):       po y                  *
*  Simulation Length:               sl 1200ns             *
*                                                         *
***********************************************************
*
cl reset 1200ns 01111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf ldins.ClMi
ti LOAD Immediate
po y
*
***********************************************************
*                                                         *
*  Power given after simulation:                          *
*                                                         *
*  Average Power:        Not Recorded                     *
*  Average Current:      Not Recorded                     *
*                                                         *
***********************************************************
```

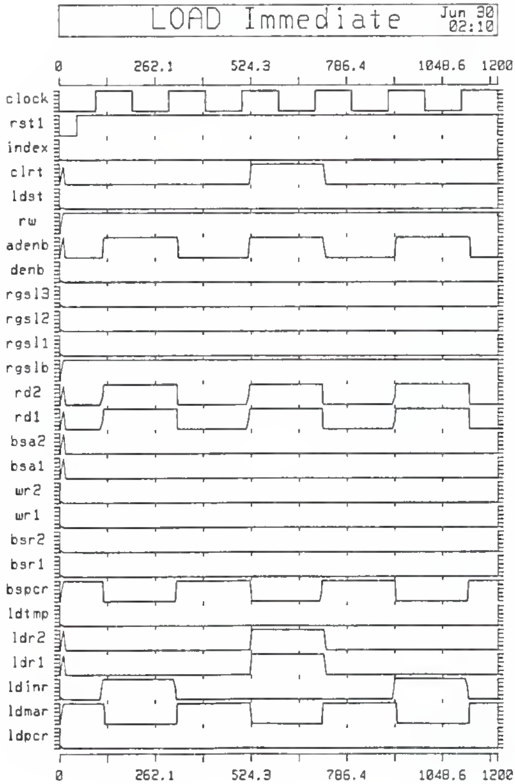

Figure B28: Plot Of Results From Load Immediate Instruction Simulation.

```
***********************************************************
*                                                         *
*           Simulation File for checking TORO             *
*           store instruction control signal              *
*          generation during indexed addressing.          *
*                       CNTL.SIM4                         *
*                                                         *
***********************************************************
*
***********************************************************
*                                                         *
*  Setting the instruction class and addressing mode      *
*  inputs to load and indexed. Setting the clock to       *
*  5 MHz.                                                 *
*                                                         *
***********************************************************
*
hi m1 m0 c0
lo c1
cl clock 200ns 01
as clock clock
*
***********************************************************
*                                                         *
*  These are the names of the control                     *
*  signals that drive the control signal                  *
*  buffers:                                               *
*                                                         *
*  pl ldpc ldmr T1 ldr ldst ldtp clrt bspc busr           *
*  pl balu read index rgsel denb adenb                    *
*                                                         *
*  and the following are the corresponding               *
*  outputs driver signal names:                           *
*                                                         *
***********************************************************
*
pl ldpcr ldmar ldinr ldr1 ldr2 ldtmp bspcr bsr1 bsr2
pl bsa1 bsa2 rd1 rd2 wr1 rgslb rgsl1 rgsl2 rgsl3
*
***********************************************************
*                                                         *
*  These signals are for driving the pad                  *
*  enables and the R/W signal to the peripherals:         *
*                                                         *
***********************************************************
*
pl denb adenb rw
*
```

```
************************************************************
*                                                          *
*  These signals are controls, but internal                *
*  to the CNTL functional block:                           *
*                                                          *
************************************************************
*
pl ldst clrt index
*
************************************************************
*                                                          *
*  Simulation Parameters:                                  *
*                                                          *
*  Plot Step:                        ps 10ns               *
*  Power Output? ( y = yes ):        po y                  *
*  Simulation Length:                sl 1200ns             *
*                                                          *
************************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf stins.CsMx
ti STORE Indexed
po y
*
************************************************************
*                                                          *
*  Power given after simulation:                           *
*                                                          *
*  Average Power:       1.83236 mW                         *
*  Average Current:     0.366472 mA                        *
*                                                          *
************************************************************
```

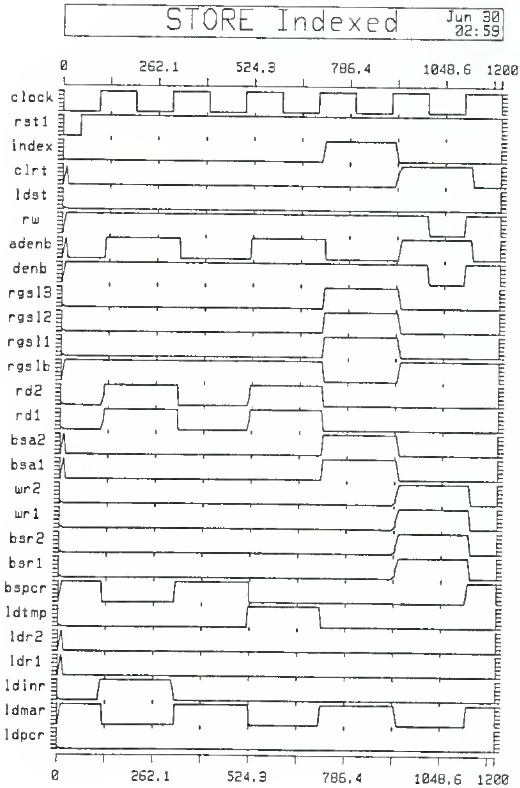

Figure B29: Plot Of Results From STORE Indexed Instruction Simulation

```
****************************************************************
*                                                              *
*          Simulation File for checking TORO                  *
*          store instruction control signal                   *
*         generation during direct addressing.                *
*                         CNTL.SIM5                            *
*                                                              *
****************************************************************
*
****************************************************************
*                                                              *
*  Setting the instruction class and addressing mode          *
*  inputs to load and indexed. Setting the clock to           *
*  5 MHz.                                                      *
*                                                              *
****************************************************************
*
hi m1 c0
lo m0 c1
cl clock 200ns 01
as clock clock
*
****************************************************************
*                                                              *
*  These are the names of the control                         *
*  signals that drive the control signal                      *
*  buffers:                                                    *
*                                                              *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr               *
*  pl balu read index rgsel denb adenb                        *
*                                                              *
*  and the follcwing are the corresponding                    *
*  outputs driver signal names:                               *
*                                                              *
****************************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsl1 rgsl2 rgsl3
*
****************************************************************
*                                                              *
*  These signals are for driving the pad                      *
*  enables and the R/W signal to the peripherals:             *
*                                                              *
****************************************************************
*
pl denb adenb rw
*
```

```
**************************************************************
*                                                            *
*  These signals are controls, but internal                  *
*  to the CNTL functional block:                             *
*                                                            *
**************************************************************
*
pl ldst clrt index
*
**************************************************************
*                                                            *
*  Simulation Parameters:                                    *
*                                                            *
*  Plot Step:                         ps 10ns                *
*  Power Output? ( y = yes ):         po y                   *
*  Simulation Length:                 sl 1200ns              *
*                                                            *
**************************************************************
*
cl reset 1200ns 0111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf stins.ClMd
ti STORE Direct
po y
*
**************************************************************
*                                                            *
*  Power given after simulation:                             *
*                                                            *
*  Average Power:    1.79657 mW                              *
*  Average Current:  0.324085 mA                             *
*                                                            *
**************************************************************
```

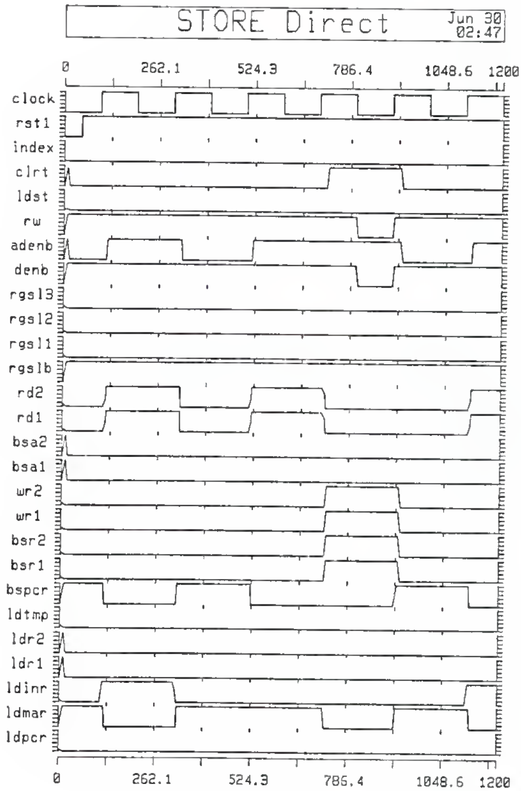

Figure B30: Plot Of Results From STORE Direct Instruction Simulation.

```
*************************************************************
*                                                           *
*             Simulation File for checking TORO             *
*              ALU instruction control signal               *
*            generation during indexed addressing.          *
*                         CNTL.SIM7                         *
*                                                           *
*************************************************************
*
*************************************************************
*                                                           *
*  Setting the instruction class and addressing mode        *
*  inputs to load and indexed. Setting the clock to         *
*  5 MHz.                                                   *
*                                                           *
*************************************************************
*
hi ml m0 c0 cl
cl clock 200ns 01
as clock clock
*
*************************************************************
*                                                           *
*  These are the names of the control                       *
*  signals that drive the control signal                    *
*  buffers:                                                 *
*                                                           *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr             *
*  pl balu read index rgsel denb adenb                      *
*                                                           *
*  and the following are the corresponding                  *
*  outputs driver signal names:                             *
*                                                           *
*************************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 lctmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
*************************************************************
*                                                           *
*  These signals are for driving the pad                    *
*  enables and the R/W signal to the peripherals:           *
*                                                           *
*************************************************************
*
pl denb adenb rw
*
```

```
*****************************************************************
*                                                               *
*  These signals are controls, but internal                     *
*  to the CNTL functional block:                                *
*                                                               *
*****************************************************************
*
pl ldst clrt index
*
*****************************************************************
*                                                               *
*  Simulation Parameters:                                       *
*                                                               *
*  Plot Step:                          ps 10ns                  *
*  Power Output? ( y = yes ):          po y                     *
*  Simulation Length:                  sl 1200ns                *
*                                                               *
*****************************************************************
*
cl reset 1200ns 0111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf aluins.CaMx
ti ALU Indexed
po y
*
*****************************************************************
*                                                               *
*  Power given after simulation:                                *
*                                                               *
*  Average Power:     1.70068 mW                                *
*  Average Current:   0.340135 mA                               *
*                                                               *
*****************************************************************
```

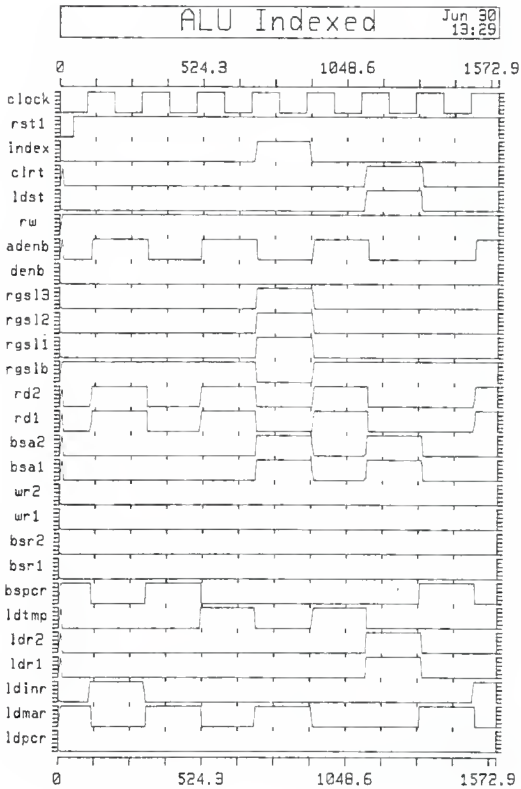

Figure 31: Plot Of Results From ALU Indexed Instruction Simulation

```
***********************************************************
*                                                         *
*          Simulation File for checking TORO              *
*           ALU instruction control signal                *
*         generation during direct addressing.            *
*                     CNTL.SIM8                           *
*                                                         *
***********************************************************
*
***********************************************************
*                                                         *
*  Setting the instruction class and addressing mode      *
*  inputs to load and indexed. Setting the clock to       *
*  5 MHz.                                                 *
*                                                         *
***********************************************************
*
hi m1 c0 c1
lo m0
cl clock 200ns 01
as clock clock
*
***********************************************************
*                                                         *
*  These are the names of the control                     *
*  signals that drive the control signal                  *
*  buffers:                                               *
*                                                         *
*  pl ldpc ldmr T1 ldr ldst ldtp clrt bspc busr           *
*  pl balu read index rgsel denb adenb                    *
*                                                         *
*  and the following are the corresponding                *
*  outputs driver signal names:                           *
*                                                         *
***********************************************************
*
pl ldpcr ldmar ldinr ldr1 ldr2 ldtmp bspcr bsr1 bsr2
pl bsa1 bsa2 rd1 rd2 wr1 rgs1b rgs11 rgs12 rgs13
*
***********************************************************
*                                                         *
*  These signals are for driving the pad                  *
*  enables and the R/W signal to the peripherals:         *
*                                                         *
***********************************************************
*
pl denb adenb rw
*
```

```
*********************************************************
*                                                       *
*  These signals are controls, but internal             *
*  to the CNTL functional block:                        *
*                                                       *
*********************************************************
*
pl ldst clrt index
*
*********************************************************
*                                                       *
*  Simulation Parameters:                               *
*                                                       *
*  Plot Step:                      ps 10ns              *
*  Power Output? ( y = yes ):      po y                 *
*  Simulation Length:              sl 1200ns            *
*                                                       *
*********************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf aluins.CaMd
ti ALU Direct
po y
*
*********************************************************
*                                                       *
*  Power given after simulation:                        *
*                                                       *
*  Average Power:    1.72619 mW                         *
*  Average Current:  0.345238 mA                        *
*                                                       *
*********************************************************
```

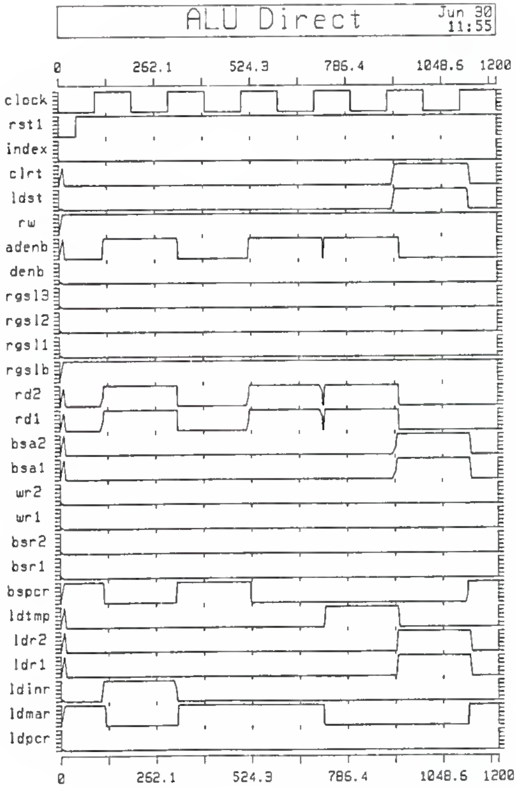

Figure B32: Plot Of Results From ALU Direct Instruction Simulation.

```
**********************************************************
*                                                        *
*          Simulation File for checking TORO             *
*           ALU instruction control signal               *
*         generation during immediate addressing.        *
*                      CNTL.SIM9                         *
*                                                        *
**********************************************************
*
**********************************************************
*                                                        *
*  Setting the instruction class and addressing mode     *
*  inputs to load and indexed. Setting the clock to      *
*  5 MHz.                                                *
*                                                        *
**********************************************************
*
hi m0 c0 c1
lo m1
cl clock 200ns 01
as clock clock
*
**********************************************************
*                                                        *
*  These are the names of the control                    *
*  signals that drive the control signal                 *
*  buffers:                                              *
*                                                        *
*  pl ldpc ldmr T1 ldr ldst ldtp clrt bspc busr          *
*  pl balu read index rgsel denb adenb                   *
*                                                        *
*  and the following are the corresponding               *
*  outputs driver signal names:                          *
*                                                        *
**********************************************************
*
pl ldpcr ldmar ldinr ldr1 ldr2 ldtmp bspcr bsr1 bsr2
pl bsa1 bsa2 rd1 rd2 wr1 rgs1b rgs11 rgs12 rgs13
*
**********************************************************
*                                                        *
*  These signals are for driving the pad                 *
*  enables and the R/W signal to the peripherals:        *
*                                                        *
**********************************************************
*
pl denb adenb rw
*
```

```
************************************************************
*                                                          *
*  These signals are controls, but internal                *
*  to the CNTL functional block:                           *
*                                                          *
************************************************************
*
pl ldst clrt index
*
************************************************************
*                                                          *
*  Simulation Parameters:                                  *
*                                                          *
*  Plot Step:                       ps 10ns                *
*  Power Output? ( y = yes ):       po y                   *
*  Simulation Length:               sl 1200ns              *
*                                                          *
************************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf aluins.CaMi
ti ALU Immediate
po y
*
************************************************************
*                                                          *
*  Power given after simulation:                           *
*                                                          *
*  Average Power:     1.65797 mW                           *
*  Average Current:  0.331594 mA                           *
*                                                          *
************************************************************
```

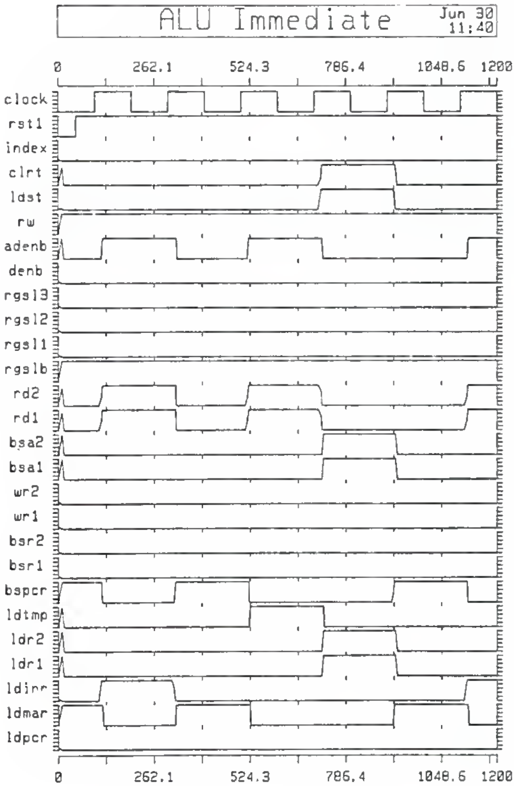

Figure B33: Plot Of Results From ALU Immediate Instruction Simulation.

```
**********************************************************
*                                                        *
*           Simulation File for checking TORO            *
*            ALU instruction control signal              *
*         generation during inherent addressing.         *
*                      CNTL.SIM10                        *
*                                                        *
**********************************************************
*
**********************************************************
*                                                        *
*  Setting the instruction class and addressing mode     *
*  inputs to load and indexed. Setting the clock to      *
*  5 MHz.                                                *
*                                                        *
**********************************************************
*
hi c0 cl
lo ml m0
cl clock 200ns 01
as clock clock
*
**********************************************************
*                                                        *
*  These are the names of the control                    *
*  signals that drive the control signal                 *
*  buffers:                                              *
*                                                        *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr          *
*  pl balu read index rgsel denb adenb                   *
*                                                        *
*  and the following are the corresponding               *
*  outputs driver signal names:                          *
*                                                        *
**********************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
**********************************************************
*                                                        *
*  These signals are for driving the pad                 *
*  enables and the R/W signal to the peripherals:        *
*                                                        *
**********************************************************
*
pl denb adenb rw
*
```

```
*************************************************************
*                                                           *
*  These signals are controls, but internal                 *
*  to the CNTL functional block:                            *
*                                                           *
*************************************************************
*
pl ldst clrt index
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                          ps 10ns              *
*  Power Output? ( y = yes ):          po y                 *
*  Simulation Length:                  sl 1200ns            *
*                                                           *
*************************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf aluins.CaMh
ti ALU Inherent
po y
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:     1.72431 mW                            *
*  Average Current:   0.34861 mA                            *
*                                                           *
*************************************************************
```

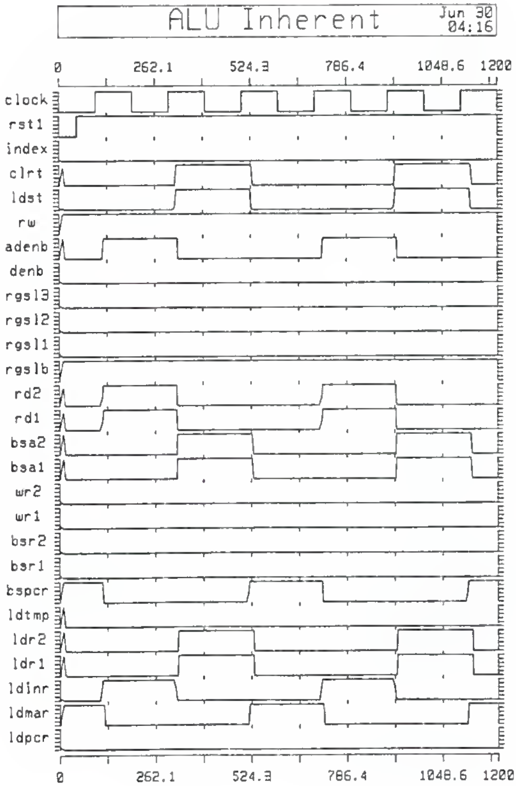

Figure B34: Plot Of Results From ALU Inherent Instruction Simulation.

In the following three tests, the control signal generation for the branch instruction was performed. In these tests, it was assumed that the status register was decoded to give an inactive branch control signal. Thus, for the three following tests, the loading of the program counter was inhibited.

BRANCH Immediate: This simulation checked for correct BRANCH instruction control signal generation for immediate addressing. For this test, the branch control signal was forced low, inhibiting program counter loading.

BRANCH Direct: This simulation checked for correct BRANCH instruction control signal generation for direct addressing. For this test, the branch control signal was forced low, inhibiting program counter loading.

BRANCH Indexed: This simulation checked for correct BRANCH instruction control signal generation for indexed addressing. For this test, the branch control signal was forced low, inhibiting program counter loading.

```
***********************************************************
*                                                         *
*          Simulation File for checking TORO              *
*           BRANCH instruction control signal             *
*          generation during immediate addressing.        *
*                        CNTL.SIM11                       *
*                                                         *
***********************************************************
*
***********************************************************
*                                                         *
*  Setting the instruction class and addressing mode      *
*  inputs to load and indexed. Setting the clock to       *
*  5 MHz. For this simulation, the branch control         *
*  signal is tied LOW.                                    *
*                                                         *
***********************************************************
*
hi m0 cl
lo ml c0 br s2 sl s0
cl clock 200ns 01
as clock clock
*
***********************************************************
*                                                         *
*  These are the names of the control                     *
*  signals that drive the control signal                  *
*  buffers:                                               *
*                                                         *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr           *
*  pl balu read index rgsel denb adenb                    *
*                                                         *
*  and the following are the corresponding                *
*  outputs driver signal names:                           *
*                                                         *
***********************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
***********************************************************
*                                                         *
*  These signals are for driving the pad                  *
*  enables and the R/W signal to the peripherals:         *
*                                                         *
***********************************************************
*
pl denb adenb rw
*
```

```
*************************************************************
*                                                           *
*  These signals are controls, but internal                 *
*  to the CNTL functional block:                            *
*                                                           *
*************************************************************
*
pl ldst clrt index
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                         ps 10ns               *
*  Power Output? ( y = yes ):         po y                  *
*  Simulation Length:                 sl 1200ns             *
*                                                           *
*************************************************************
*
cl reset 1200ns 0111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf brins.CbMi
ti BRANCH Immediate
po y
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:    2.3221 mW                              *
*  Average Current:  0.46442 mA                             *
*                                                           *
*************************************************************
```

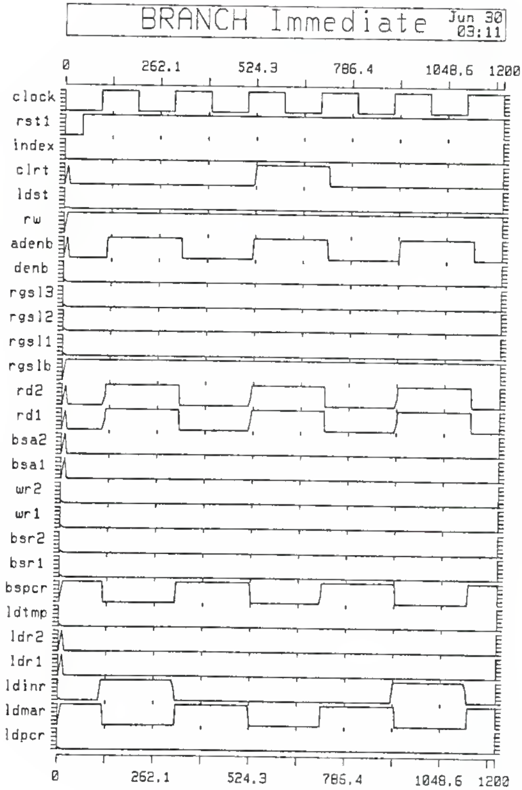

Figure B35: Plot Of Results From BRANCH Immediate Simulation - Branch Control Tied Low

```
************************************************************
*                                                          *
*           Simulation File for checking TORO              *
*            BRANCH instruction control signal             *
*          generation during direct addressing.            *
*                       CNTL.SIM12                         *
*                                                          *
************************************************************
*
************************************************************
*                                                          *
*  Setting the instruction class and addressing mode       *
*  inputs to load and indexed. Setting the clock to        *
*  5 MHz. For this simulation, the branch control          *
*  signal is tied LOW.                                     *
*                                                          *
************************************************************
*
hi m1 c1
lo m0 c0 br s2 s1 s0
cl clock 200ns 01
as clock clock
*
************************************************************
*                                                          *
*  These are the names of the control                      *
*  signals that drive the control signal                   *
*  buffers:                                                *
*                                                          *
*  pl ldpc ldmr T1 ldr ldst ldtp clrt bspc busr            *
*  pl balu read index rgsel denb adenb                     *
*                                                          *
*  and the following are the corresponding                 *
*  outputs driver signal names:                            *
*                                                          *
************************************************************
*
pl ldpcr ldmar ldinr ldr1 ldr2 ldtmp bspcr bsr1 bsr2
pl bsa1 bsa2 rd1 rd2 wr1 rgs1b rgs11 rgs12 rgs13
*
************************************************************
*                                                          *
*  These signals are for driving the pad                   *
*  enables and the R/W signal to the peripherals:          *
*                                                          *
************************************************************
*
pl denb adenb rw
*
```

```
**********************************************************
*                                                        *
*  These signals are controls, but internal              *
*  to the CNTL functional block:                         *
*                                                        *
**********************************************************
*
pl ldst clrt index
*
**********************************************************
*                                                        *
*  Simulation Parameters:                                *
*                                                        *
*  Plot Step:                          ps 10ns           *
*  Power Output? ( y = yes ):          po y              *
*  Simulation Length:                  sl 1200ns         *
*                                                        *
**********************************************************
*
cl reset 1200ns 0111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf brins.CbMd
ti BRANCH Direct
po y
*
**********************************************************
*                                                        *
*  Power given after simulation:                         *
*                                                        *
*  Average Power:    2.30768 mW                          *
*  Average Current:  0.461535 mA                         *
*                                                        *
**********************************************************
```

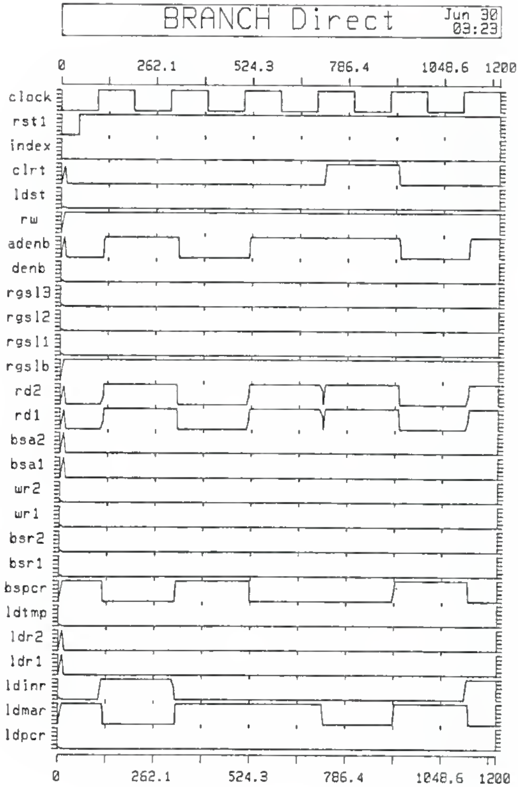

Figure B36: Plot Of Results From BRANCH Direct Simulation - Branch Signal Tied Low

```
***************************************************************
*                                                             *
*          Simulation File for checking TORO                  *
*           BRANCH instruction control signal                 *
*         generation during indexed addressing.               *
*                        CNTL.SIM13                           *
*                                                             *
***************************************************************
*
***************************************************************
*                                                             *
*  Setting the instruction class and addressing mode          *
*  inputs to load and indexed. Setting the clock to           *
*  5 MHz. For this simulation, the branch control             *
*  signal is tied LOW.                                        *
*                                                             *
***************************************************************
*
hi ml cl m0
lo c0 br s2 sl s0
cl clock 200ns 01
as clock clock
*
***************************************************************
*                                                             *
*  These are the names of the control                         *
*  signals that drive the control signal                      *
*  buffers:                                                   *
*                                                             *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr               *
*  pl balu read index rgsel denb adenb                        *
*                                                             *
*  and the following are the corresponding                    *
*  outputs driver signal names:                               *
*                                                             *
***************************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
***************************************************************
*                                                             *
*  These signals are for driving the pad                      *
*  enables and the R/W signal to the peripherals:             *
*                                                             *
***************************************************************
*
pl denb adenb rw
*
```

```
**************************************************************
*                                                            *
*  These signals are controls, but internal                  *
*  to the CNTL functional block:                             *
*                                                            *
**************************************************************
*
pl ldst clrt index
*
**************************************************************
*                                                            *
*  Simulation Parameters:                                    *
*                                                            *
*  Plot Step:                        ps 10ns                 *
*  Power Output? ( y = yes ):        po y                    *
*  Simulation Length:                sl 1200ns               *
*                                                            *
**************************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf brins.CbMx
ti BRANCH Indexed
po y
*
**************************************************************
*                                                            *
*  Power given after simulation:                             *
*                                                            *
*  Average Power:     2.52039 mW                             *
*  Average Current:   0.504078 mA                            *
*                                                            *
**************************************************************
```

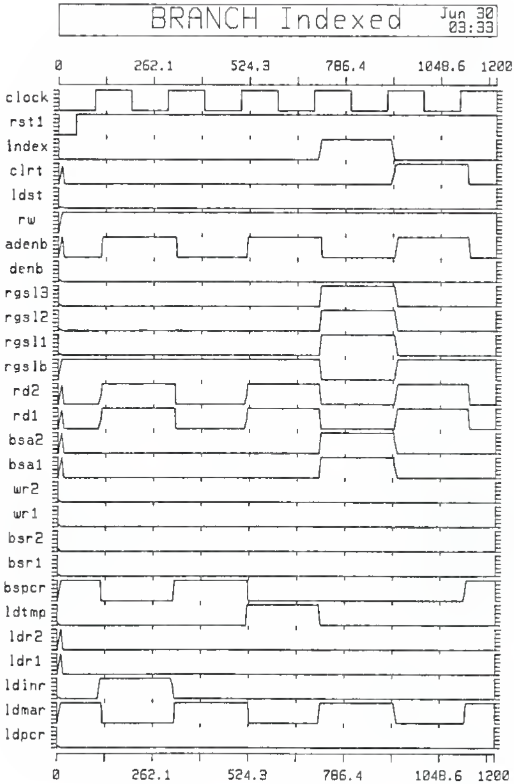

Figure B37: Plot Of Results From BRANCH Indexed Simulation - Branch Control Tied Low

In the following three tests, the control signal generation for the branch instruction was performed. In these tests, it was assumed that the status register was decoded to give an active branch control signal. Thus, for the three following tests, the loading of the program counter was not inhibited.

BRANCH Immediate: This simulation checked for correct BRANCH instruction control signal generation for immediate addressing. For this test, the branch control signal was forced high to accomplish program counter loading.

BRANCH Direct: This simulation checked for correct BRANCH instruction control signal generation for direct addressing. For this test, the branch control signal was forced high to accomplish program counter loading.

BRANCH Indexed: This simulation checked for correct BRANCH instruction control signal generation for indexed addressing. For this test, the branch control signal was forced high to accomplish program counter loading.

```
*************************************************************
*                                                           *
*           Simulation File for checking TORO               *
*            BRANCH instruction control signal              *
*          generation during immediate addressing.          *
*                        CNTL.SIM14                         *
*                                                           *
*************************************************************
*
*************************************************************
*                                                           *
*  Setting the instruction class and addressing mode        *
*  inputs to load and indexed. Setting the clock to         *
*  5 MHz. For this simulation, the branch control           *
*  signal is tied HIGH.                                     *
*                                                           *
*************************************************************
*
hi cl m0 br
lo m1 c0 s2 s1 s0
cl clock 200ns 01
as clock clock
*
*************************************************************
*                                                           *
*  These are the names of the control                       *
*  signals that drive the control signal                    *
*  buffers:                                                 *
*                                                           *
*  pl ldpc ldmr T1 ldr ldst ldtp clrt bspc busr             *
*  pl balu read index rgsel denb adenb                      *
*                                                           *
*  and the following are the corresponding                  *
*  outputs driver signal names:                             *
*                                                           *
*************************************************************
*
pl ldpcr ldmar ldinr ldr1 ldr2 ldtmp bspcr bsr1 bsr2
pl bsa1 bsa2 rd1 rd2 wr1 rgs1b rgs11 rgs12 rgs13
*
*************************************************************
*                                                           *
*  These signals are for driving the pad                    *
*  enables and the R/W signal to the peripherals:           *
*                                                           *
*************************************************************
*
pl denb adenb rw
*
```

```
***********************************************************
*                                                         *
*  These signals are controls, but internal               *
*  to the CNTL functional block:                          *
*                                                         *
***********************************************************
*
pl ldst clrt index
*
***********************************************************
*                                                         *
*  Simulation Parameters:                                 *
*                                                         *
*  Plot Step:                        ps 10ns              *
*  Power Output? ( y = yes ):        po y                 *
*  Simulation Length:                sl 1200ns            *
*                                                         *
***********************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf BRins.CbMi
ti BRANCH Immediate
po y
*
***********************************************************
*                                                         *
*  Power given after simulation:                          *
*                                                         *
*  Average Power:     1.57924 mW                          *
*  Average Current:   0.315849 mA                         *
*                                                         *
***********************************************************
```

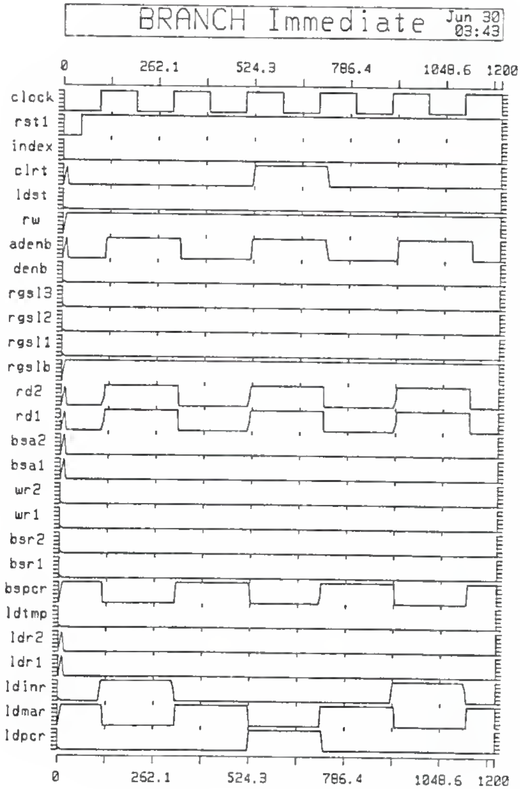

Figure B38: Plot Of Results From BRANCH Immediate Simulation - Branch Control Tied High

```
*************************************************************
*                                                           *
*           Simulation File for checking TORO               *
*            BRANCH instruction control signal              *
*          generation during direct addressing.             *
*                       CNTL.SIM15                          *
*                                                           *
*************************************************************
*
*************************************************************
*                                                           *
*  Setting the instruction class and addressing mode        *
*  inputs to load and indexed. Setting the clock to         *
*  5 MHz. For this simulation, the branch control           *
*  signal is tied HIGH.                                     *
*                                                           *
*************************************************************
*
hi ml cl br
lo m0 c0 s2 sl s0
cl clock 200ns 01
as clock clock
*
*************************************************************
*                                                           *
*  These are the names of the control                       *
*  signals that drive the control signal                    *
*  buffers:                                                 *
*                                                           *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr             *
*  pl balu read index rgsel denb adenb                      *
*                                                           *
*  and the following are the corresponding                  *
*  outputs driver signal names:                             *
*                                                           *
*************************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
*************************************************************
*                                                           *
*  These signals are for driving the pad                    *
*  enables and the R/W signal to the peripherals:           *
*                                                           *
*************************************************************
*
pl denb adenb rw
*
```

```
**********************************************************
*                                                        *
*  These signals are controls, but internal              *
*  to the CNTL functional block:                         *
*                                                        *
**********************************************************
*
pl ldst clrt index
*
**********************************************************
*                                                        *
*  Simulation Parameters:                                *
*                                                        *
*  Plot Step:                       ps 10ns              *
*  Power Output? ( y = yes ):       po y                 *
*  Simulation Length:               sl 1200ns            *
*                                                        *
**********************************************************
*
cl reset 1200ns 0111111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf BRins.CbMd
ti BRANCH Direct
po y
*
**********************************************************
*                                                        *
*  Power given after simulation:                         *
*                                                        *
*  Average Power:     1.58795 mW                         *
*  Average Current:   0.317591 mA                        *
*                                                        *
**********************************************************
```

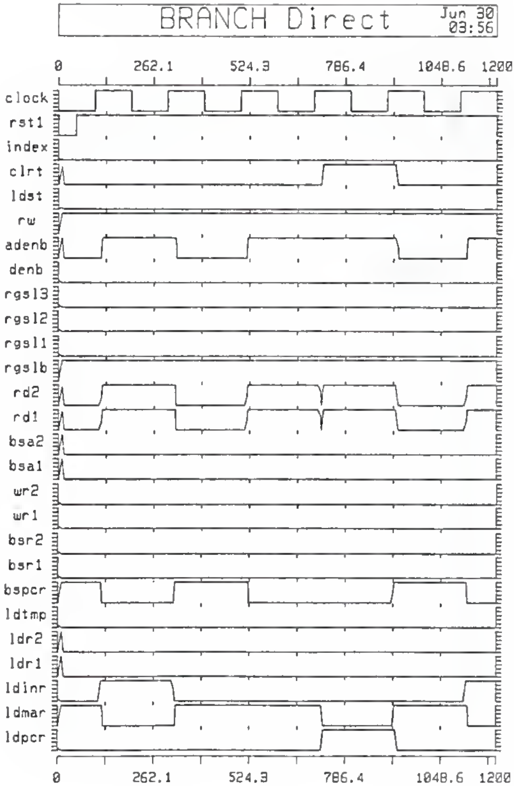

Figure B39: Plot Of Results From BRANCH Direct Simulation - Branch Control Tied High

```
*************************************************************
*                                                           *
*          Simulation File for checking TORO                *
*           BRANCH instruction control signal               *
*         generation during indexed addressing.             *
*                       CNTL.SIM16                          *
*                                                           *
*************************************************************
*
*************************************************************
*                                                           *
*  Setting the instruction class and addressing mode        *
*  inputs to branch and indexed. Setting the clock to       *
*  5 MHz. For this simulation, the branch control           *
*  signal is tied HIGH.                                     *
*                                                           *
*************************************************************
*
hi ml cl m0 br
lo c0 s2 sl s0
cl clock 200ns 01
as clock clock
*
*************************************************************
*                                                           *
*  These are the names of the control                       *
*  signals that drive the control signal                    *
*  buffers:                                                 *
*                                                           *
*  pl ldpc ldmr Tl ldr ldst ldtp clrt bspc busr             *
*  pl balu read index rgsel denb adenb                      *
*                                                           *
*  and the following are the corresponding                  *
*  outputs driver signal names:                             *
*                                                           *
*************************************************************
*
pl ldpcr ldmar ldinr ldrl ldr2 ldtmp bspcr bsrl bsr2
pl bsal bsa2 rdl rd2 wrl rgslb rgsll rgsl2 rgsl3
*
*************************************************************
*                                                           *
*  These signals are for driving the pad                    *
*  enables and the R/W signal to the peripherals:           *
*                                                           *
*************************************************************
*
pl denb adenb rw
*
```

```
*************************************************************
*                                                           *
*  These signals are controls, but internal                 *
*  to the CNTL functional block:                            *
*                                                           *
*************************************************************
*
pl ldst clrt index
*
*************************************************************
*                                                           *
*  Simulation Parameters:                                   *
*                                                           *
*  Plot Step:                        ps 10ns                *
*  Power Output? ( y = yes ):        po y                   *
*  Simulation Length:                sl 1200ns              *
*                                                           *
*************************************************************
*
cl reset 1200ns 0111111111111111111111
as reset rstl
pl rstl clock
sl 1200ns
ps 10ns
cm + hpr
pf BRins.CbMx
ti BRANCH Indexed
po y
*
*************************************************************
*                                                           *
*  Power given after simulation:                            *
*                                                           *
*  Average Power:     1.81342 mW                            *
*  Average Current:   0.362683 mA                           *
*                                                           *
*************************************************************
```

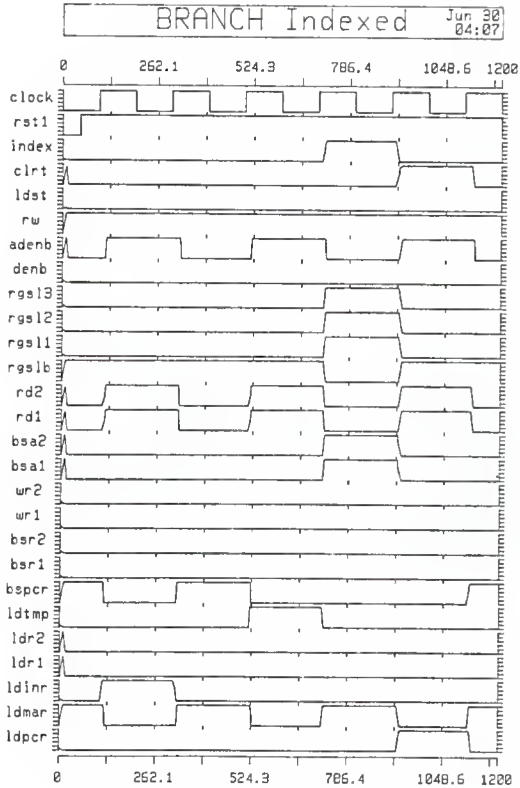

Figure B40: Plot Of Results From BRANCH Indexed Simulation - Branch Control Tied High

The following two tests were performed to check the control logic for proper ALU control signal generation for the four addressing modes. During inherent, direct, and immediate addressing, the ALU was allowed to function as expected from the decoding of the instruction register. However, for the indexed addressing mode, the ALU was forced to perform an ADD for the generation of the indexed address.

ALU Control1: This simulation checked for correct ALU instruction control signal generation for the ALU. In this simulation, the ALU signals were checked for all addressing modes except indexed addressing.

ALU Control2: This simulation checked for correct instruction control signal generation for the ALU. In this simulation, the ALU signals were checked for the indexed addressing mode.

```
************************************************************
*                                                          *
*              Simulation file for testing                 *
*                    the ALU control.                      *
*                    CNTL.ALU.SIM17                        *
*                                                          *
************************************************************
*
*  This file was updated 2/19/89 to check
*  CNTL alu control after extensive modification
*  to the ALU.
*
*  Updated after adding a control signal
*  for loading the inverse of the carry
*  in during subtraction.
*
************************************************************
*                                                          *
*  These are the clock signals to drive the                *
*  Instruction Register inputs and the index               *
*  control output to check for correct ALU                 *
*  control signal generation. This is for                  *
*  index forced LOW.                                       *
*                                                          *
************************************************************
*
cl mh 1600ns    0000000011111111
as mh mh
cl s2 1600ns    0000111100001111
as s2 s2
cl s1 1600ns    0011001100110011
as s1 s1
cl s0 1600ns    0101010101010101
as s0 s0
cl index 1600ns 0000000000000000
as index index
*
```

```
**************************************************************
*                                                            *
*  These are the names of the control signals                *
*  that drive the output buffers:                            *
*                                                            *
*  pl cIn tzro tone aone sax0 sax1                           *
*  pl szro salu0 salu1 selc                                  *
*                                                            *
*  and these are the output driver's names:                  *
*                                                            *
**************************************************************
*
pl cin tZRO tONE aONE
pl s01 s02 sb01 sb02 s11 s12 sb11 sb12
pl sZRO
pl x01 x02 xb01 xb02 x11 x12 xb11 xb12
*
**************************************************************
*                                                            *
*  These ALU control signals have no external                *
*  buffers to drive. They drive gates internal               *
*  to CNTL. cslt0 and clstl are rol and ror,                 *
*  respectfully.                                             *
*                                                            *
**************************************************************
*
pl selc ror rol cinv cm
*
```

```
***********************************************************
*                                                         *
*  Simulation Parameters:                                 *
*                                                         *
*  Plot Step:                           ps 10ns           *
*  Power Output? ( y = yes ):           po y              *
*  Simulation Length:                   sl 1600ns         *
*                                                         *
***********************************************************
*
sl 1600ns
ps 10ns
cm + hpr
ti ALU Control
pf CNTL.alu.outl
po y
*
***********************************************************
*                                                         *
*  Power given after simulation:                          *
*                                                         *
*  Average Power:    Not Recorded                         *
*  Average Current:  Not Recorded                         *
*                                                         *
***********************************************************
```

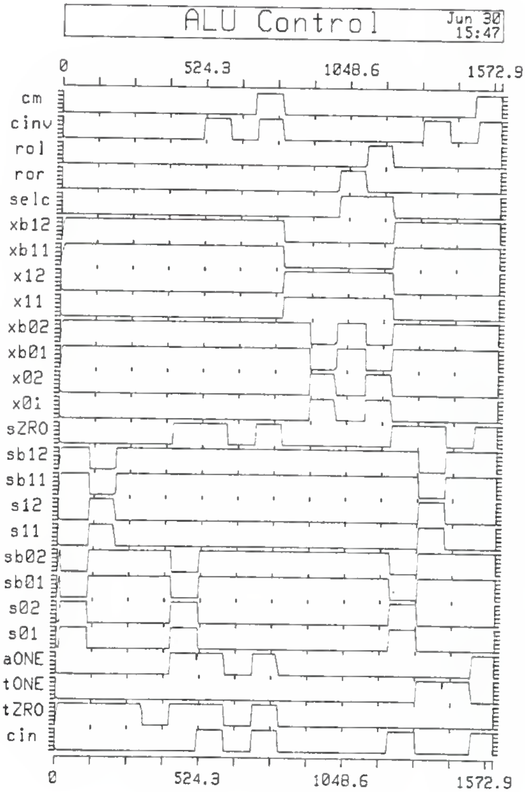

Figure B41: Plot Of Results From ALU Control Signal Simulation - Inh/Imm/Direct Addressing

```
************************************************************
*                                                          *
*              Simulation file for testing                 *
*                  the ALU control.                        *
*                  CNTL.ALU.SIM18                          *
*                                                          *
************************************************************
*
*  This file was updated 2/19/89 to check
*  CNTL alu control after extensive modification
*  to the ALU.
*
*  Updated after adding a control signal
*  for loading the inverse of the carry
*  in during subtraction.
*
************************************************************
*                                                          *
*  These are the clock signals to drive the                *
*  Instruction Register inputs and the index               *
*  control output to check for correct ALU                 *
*  control signal generation. This is for                  *
*  index forced HIGH.                                      *
*                                                          *
************************************************************
*
cl mh 1600ns    0000000011111111
as mh mh
cl s2 1600ns    0000111100001111
as s2 s2
cl s1 1600ns    0011001100110011
as s1 s1
cl s0 1600ns    0101010101010101
as s0 s0
cl index 1600ns 1111111111111111
as index index
*
```

```
***********************************************************
*                                                         *
*  These are the names of the control signals             *
*  that drive the output buffers:                         *
*                                                         *
*  pl cIn tzro tone aone sax0 sax1                        *
*  pl szro salu0 salu1 selc                               *
*                                                         *
*  and these are the output driver's names:               *
*                                                         *
***********************************************************
*
pl cin tZRO tONE aONE
pl s01 s02 sb01 sb02 s11 s12 sb11 sb12
pl sZRO
pl x01 x02 xb01 xb02 x11 x12 xb11 xb12
*
***********************************************************
*                                                         *
*  These ALU control signals have no external             *
*  buffers to drive. They drive gates internal            *
*  to CNTL. cslt0 and clstl are rol and ror,              *
*  respectfully.                                          *
*                                                         *
***********************************************************
*
pl selc ror rol cinv cm
*
```

```
************************************************************
*                                                          *
*  Simulation Parameters:                                  *
*                                                          *
*  Plot Step:                           ps 10ns            *
*  Power Output? ( y = yes ):           po y               *
*  Simulation Length:                   sl 1600ns          *
*                                                          *
************************************************************
*
sl 1600ns
ps 10ns
cm + hpr
ti ALU Control
pf CNTL.alu.out2
po y
*
************************************************************
*                                                          *
*  Power given after simulation:                           *
*                                                          *
*  Average Power:    2.69981 mW                            *
*  Average Current:  2.67346 mA                            *
*                                                          *
************************************************************
```

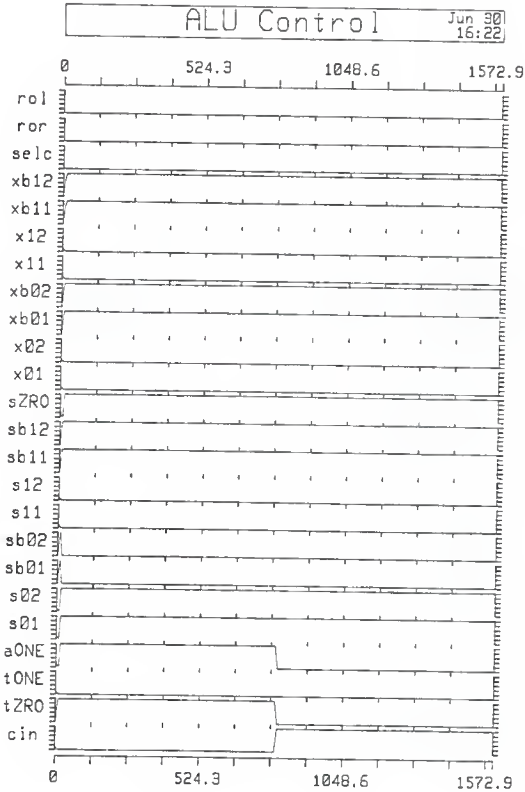

Figure B42: Plot Of Results From ALU Control Signal Simulation - Indexed Addressing

For the following test, the carry decode circuitry was tested for proper carry register input generation. Recall that the carry register has four possible inputs: one from the most significant bit for a roll left, one from the least significant bit for a roll left, one from the inverted carry out from the ALU during a subtract operation, and one for the non-inverted carry out from the ALU during all other operations.

CNTL Carry Decode: Simulation to check the carry register input loading from the ALU and output multiplexing to the ALU.

The following test was for the branch decode circuitry. Recall that the branch control signal was generated based on the outputs from the instruction register and the contents of the status register.

CNTL Branch Decode: This simulation was performed to check for proper branch decode signal generation. This test is not exhaustive.

```
*************************************************************
*                                                           *
*      Simulation file for checking the loading and         *
*        multiplexing of the carry input. The other         *
*               status bits are also loaded.                *
*                     CNTL.CARRY.SIM                        *
*                                                           *
*************************************************************
*
*************************************************************
*                                                           *
*  Starting the clock for ten cycles:                       *
*                                                           *
*************************************************************
*
cl clock 1000ns 0011001100110011001100110011001100110011
as clock clock
pl clock
*
*************************************************************
*                                                           *
*  Let the registers always be enabled:                     *
*                                                           *
*************************************************************
*
hi ldst
*
*************************************************************
*                                                           *
*  Generating a carry input, almost at random. The          *
*  cinv control signal comes from the ALU instruction       *
*  generation. It is high for SUB, CMP, TST, and DEC        *
*  instructions.                                            *
*                                                           *
*************************************************************
*
cl carry 1000ns 0110000001111111000000111100110011111000
as carry carry
cl cinv  1000ns 0000000000000000111111111111111100000000
as cinv cinv
pl carry cinv
*
```

```
************************************************************
*                                                          *
*  Selecting a multiplexer input. The carry bit input      *
*  multiplexer is controlled with the ROR and ROL          *
*  ALU instructions. Note that they are mutually           *
*  exclusive:                                              *
*                                                          *
************************************************************
*
cl csltl 1000ns 0000000000000000011111111111111110000111
as csltl ror
cl cslt0 1000ns 0000000001111111100000000000000000000000
as cslt0 rol
pl ror rol
*
************************************************************
*                                                          *
*  Creating the shift inputs, again, at random. These      *
*  inputs come from the MSB and LSB of the output of       *
*  the ALU.                                                *
*                                                          *
************************************************************
*
cl shlin 1000ns 1111100000011110000000011111111100000000
as shlin shlin
cl shrin 1000ns 0000000001111111100000000111111100000000
as shrin shrin
pl shlin shrin
*
************************************************************
*                                                          *
*  Loading the other status bits:                          *
*                                                          *
************************************************************
*
cl neg   1000ns 0000000000000111111110000111100001111000
as neg neg
cl zero  1000ns 0000011110000111100000000111100001111000
as zero zero
pl neg zero ndout cdout zdout
```

```
**********************************************************
*                                                        *
*  Enabling the carry output to the ALU. Recall that     *
*  the output of the carry bit status register is a      *
*  single AND gate. It passes the carry bit onto the     *
*  ALU only during roll operation. Otherwise, the        *
*  carry bit into the ALU is zero, as for shift          *
*  operations.                                           *
*                                                        *
**********************************************************
*
cl selc 1000ns 11111111111111110000000000001111111111
as selc selc
pl selc shrlo
*
**********************************************************
*                                                        *
*  Simulation Parameters:                                *
*                                                        *
*  Plot Step:                              ps 10ns       *
*  Power Output? ( y = yes ):              po y          *
*  Simulation Length:                      sl 1000ns     *
*                                                        *
**********************************************************
ps 10ns
po y
sl 1000ns
ti CNTL Carry Decode
pf CNTL.carry.out
cm + hlpl
*
**********************************************************
*                                                        *
*  Power given after simulation:                         *
*                                                        *
*  Average Power:    9.1631 mW                           *
*  Average Current:  1.83262 mA                          *
*                                                        *
**********************************************************
```

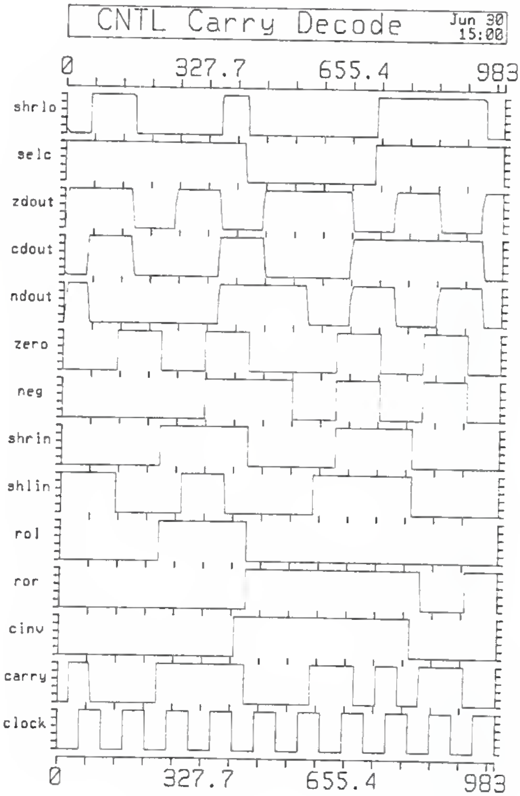

Figure B43: Plot Of Results From Carry Decode And Status Register Loading Simulation.

```
***********************************************************
*                                                         *
*       Simulation file to test the branch control        *
*                  signal generation.                     *
*                   CNTL.BRANCH.SIM                       *
*                                                         *
***********************************************************
*
***********************************************************
*                                                         *
*  The following inputs are from the instruction          *
*  register. They are decoded to discover what branch     *
*  conditions are required to allow the branch control    *
*  signal to go high.                                     *
*                                                         *
***********************************************************
*
cl s2 1600ns 0000111100001111
as s2 s2
cl s1 1600ns 0011001100110011
as s1 s1
cl s0 1600ns 0101010101010101
as s0 s0
pl s2 s1 s0
*
***********************************************************
*                                                         *
*  Inputs into the branch decode circuit. These inputs    *
*  normally come from the status register. For simu-      *
*  lation, these inputs were forced.                      *
*                                                         *
***********************************************************
*
cl ndout 1600ns 1111111100000000
as ndout ndout
cl cdout 1600ns 1111111100000000
as cdout cdout
cl zdout 1600ns 0000000011111111
as zdout zdout
*
***********************************************************
*                                                         *
*  Plotting the nodes:                                    *
*                                                         *
***********************************************************
*
pl ndout zdout cdout
pl br
*
```

```
*************************************************************
*                                                           *
*   Simulations Parameters:                                 *
*                                                           *
*   Plot Step:                               ps 10ns        *
*   Power Output? ( y = yes ):               po y           *
*   Simulation Length:                       sl 1600ns      *
*                                                           *
*************************************************************
*
po y
cm + hpr
sl 1600ns
ps 10ns
ti CNTL Branch Decode
pf CNTL.branch.out
*
*************************************************************
*                                                           *
*   Power given after simulation:                           *
*                                                           *
*   Average Power:      1.9288 mW                           *
*   Average Current:    0.38576 mA                          *
*                                                           *
*************************************************************
```

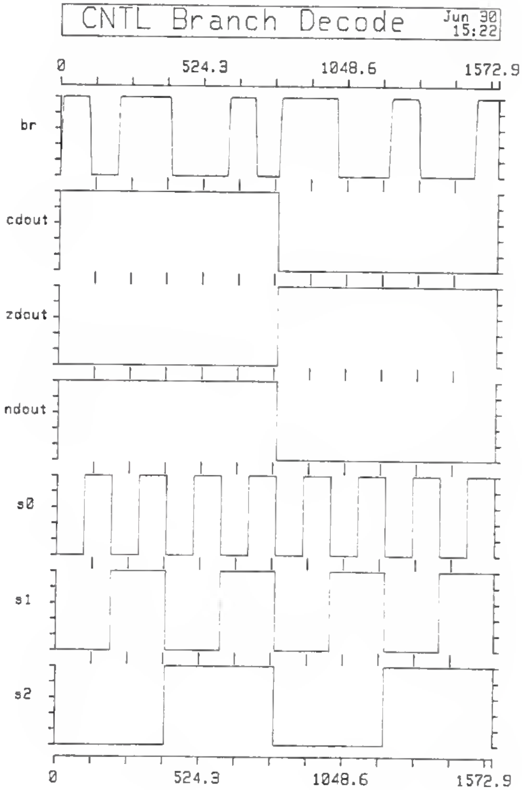

Figure B44: Plot Of Results From Branch Control Signal Decode Simulation

8.5 TORO Final Design Library

In this section, the simulation files, as well as plots and propagation delays are given. The delays were generated using the VIVID tool SIMPLOT. Admittedly, there exists some bug in SIMPLOT that cause it to often round to the nearest plot step when calculating the 10 and 90 percent values for the signals. However, a small enough plot step was used such that this rounding would not seriously affect the results extrapolated from this propagation delay data. This data was summarized in Section 4.0 above.

For each of the simulations, a large number of nodes were watched. These nodes were separated into eight groups of signals:

MAR IR TMP AX CNTL MAINBUS OUT ALU

The MAR signals were the outputs from the memory address register. The IR signals were the outputs from the instruction register. The TMP signals were the outputs from the temporary register. The AX signals were the outputs from the multiplexing of the A and X registers. The CNTL signals were the outputs from the TORO control logic. The ALU signals were the outputs from the control logic for just the ALU. The MAINBUS signals were the output data

allowed to appear on the main internal bus. The OUT signals were the outputs from the write register. There appears for each simulation described below a plot from each of the groups described above.

TORO.SIM1: For this simulation, the TORO was allowed to load immediately a data word from external memory into the accumulator, register A.

```
*************************************************************
*                                                           *
*                       TORO.SIM1                           *
*                                                           *
*        Simulation file for the LODA instruction           *
*             during immediate addressing.                  *
*                                                           *
*************************************************************
*
*  2/14/89 - Happy Valentine's Day!
*  2/16/89 - Updated after discovering
*            error in register enabling.
*  3/6/89  - Updating after discovering
*            unconnected ir bus line and
*            error in instruction generated
*            in this simulation file.
*  5/23/89 - Last set of simulations with
*            loads attached.
*
*************************************************************
*                                                           *
*  System Clock is set for 300 ns/cycle,                    *
*  or 3.33 mHz.                                             *
*                                                           *
*************************************************************
*
*                     --T1--T2--T3--T0--
cl clock  1350ns 0011001100110011001100
as clock sysclk
cl clkbar 1350ns 1100110011001100110011
as clkbar sysclkbar
cl rset   1350ns 0111111111111111111
as rset sysrs
*
*************************************************************
*                                                           *
*  Plot the address outputs:                                *
*                                                           *
*************************************************************
*
pl adr0 adr1 adr2 adr3 adr4 adr5 adr6 adr7
pl adr8 adr9 adr10 adr11 adr12 adr13 adr14 adr15
*
```

```
***********************************************************
*                                                         *
*  Address loads added here:                              *
*                                                         *
***********************************************************
*
cap adr0 1.073pf
cap adr1 1.073pf
cap adr2 1.073pf
cap adr3 1.073pf
cap adr4 1.073pf
cap adr5 1.073pf
cap adr6 1.073pf
cap adr7 1.073pf
cap adr8 1.073pf
cap adr9 1.073pf
cap adr10 1.073pf
cap adr11 1.073pf
cap adr12 1.073pf
cap adr13 1.073pf
cap adr14 1.073pf
cap adr15 1.073pf
*
***********************************************************
*                                                         *
*  Plotting the output of the Instruction                 *
*  Register, AX bus, TMP bus, and the mainbus:            *
*                                                         *
***********************************************************
*
pl ir0 ir1 ir2 ir3 ir4 ir5 ir6 ir7
pl ax0 ax1 ax2 ax3 ax4 ax5 ax6 ax7
pl ax8 ax9 ax10 ax11 ax12 ax13 ax14 ax15
pl tp0 tp1 tp2 tp3 tp4 tp5 tp6 tp7
pl tp8 tp9 tp10 tp11 tp12 tp13 tp14 tp15
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
*
***********************************************************
*                                                         *
*  Plotting the output to the data pads:                  *
*                                                         *
***********************************************************
*
pl out0 out1 out2 out3 out4 out5 out6 out7
pl out8 out9 out10 out11 out12 out13 out14 out15
*
```

```
*************************************************************
*                                                           *
*  Attaching the load for the data output here:             *
*                                                           *
*************************************************************
*
cap out0 0.665pf
cap out1 0.665pf
cap out2 0.665pf
cap out3 0.665pf
cap out4 0.665pf
cap out5 0.665pf
cap out6 0.665pf
cap out7 0.665pf
cap out8 0.665pf
cap out9 0.665pf
cap out10 0.665pf
cap out11 0.665pf
cap out12 0.665pf
cap out13 0.665pf
cap out14 0.665pf
cap out15 0.665pf
*
*************************************************************
*                                                           *
*  Plotting the TORO control signals:                       *
*                                                           *
*************************************************************
*
pl sysclk
pl ldmar ldpcr ldinr ldtmp ldrl rgsll
pl bsal bsrl bspcr rdl wrl
*
*************************************************************
*                                                           *
*  Plotting the pad enable and read/write                   *
*  signals:                                                 *
*                                                           *
*************************************************************
*
pl rw adenb denb
*
```

```
*************************************************************
*                                                           *
*  And, adding some capacitance. It turns                   *
*  out that the enable input to the pads                    *
*  is quite significant:                                    *
*                                                           *
*************************************************************
*
cap rw 0.478pf
cap adenb 5.274pf
cap denb 4.834pf
*
*************************************************************
*                                                           *
*  Plotting the alu control signals:                        *
*                                                           *
*************************************************************
*
pl aone tone tzro szro x01 x11 s01 s11
pl cin zero carry neg shrlo
*
*************************************************************
*                                                           *
*  The instruction fetched and the data                     *
*  loaded:                                                  *
*                                                           *
*************************************************************
*
*                 --T1--T2--T3--T0--
*
cl in15 1350ns    000000001111111100
as in15 in15
cl in14 1350ns    000000001111111111
as in14 in14
cl in13 1350ns    000000000000000011
as in13 in13
cl in12 1350ns    000011111111111100
as in12 in12
*
*                 --T1--T2--T3--T0--
*
cl in11 1350ns    000000000000000011
as in11 in11
cl in10 1350ns    000000000000111111
as in10 in10
cl in9  1350ns    000000001111111100
as in9 in9
cl in8 1350ns     000000001111111100
as in8 in8
```

```
*
*                   --T1--T2--T3--T0--
*
cl in7 1350ns   000000001111111100
as in7 in7
cl in6 1350ns   000000000000000000
as in6 in6
cl in5 1350ns   000000000000000000
as in5 in5
cl in4 1350ns   000011111111111111
as in4 in4
*
*                   --T1--T2--T3--T0--
*
cl in3 1350ns   000000000000000000
as in3 in3
cl in2 1350ns   000000000000000000
as in2 in2
cl in1 1350ns   000000000000111111
as in1 in1
cl in0 1350ns   000000001111111111
as in0 in0
*
*************************************************************
*                                                           *
*  The plotting parameters:                                 *
*                                                           *
*  Plot Step:                         ps 2ns                *
*  Power Output? ( y = yes ):         po y                  *
*  Simulation Length:                 sl 1350ns             *
*                                                           *
*************************************************************
*
cm - slpl
cm + hpr
pf TORO.out1
po y
sl 1350ns
ps 2ns
*
*************************************************************
*                                                           *
*  Power and Current computed after                         *
*  simulation by FACTS:                                     *
*                                                           *
*  Average Power:  13.4424 milliwatts                       *
*  Average Current:  2.68848 milliamps                      *
*                                                           *
*************************************************************
```

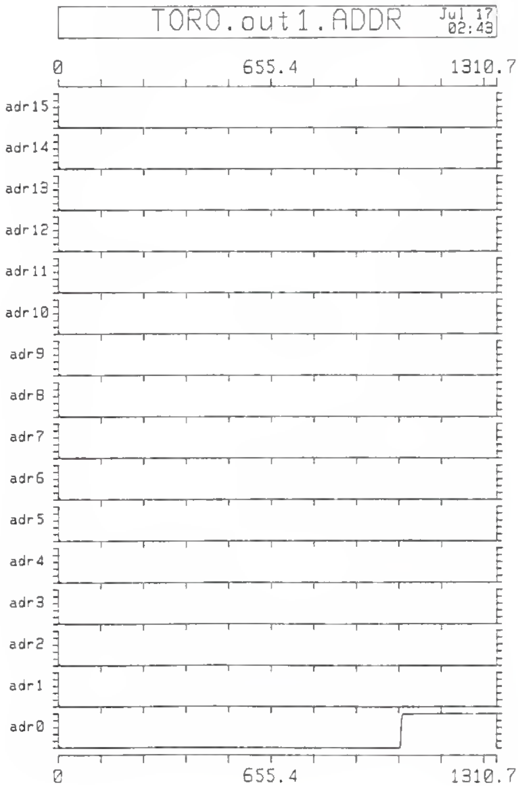

Figure B45: Plot Of Results From TORO.SIM1 Output From Memory Address Register

Propagation Delays For The TORO.SIM1 Simulation
Output From Memory Address Register

```
Running SIMPLOT 1.3
                10%              50%              90%
file  TORO.out2.addr :
    adr0:
        755.00  2.15 v
        760.00                   4.90 v           4.90 v
        1360.00 0.42 v           0.42 v           0.42 v
        1955.00 2.15 v
        1960.00                  4.90 v           4.90 v
        2560.00 0.42 v           0.42 v           0.42 v
    adr1:
        1355.00 2.15 v
        1360.00                  4.90 v           4.90 v
        2560.00 0.42 v           0.42 v           0.42 v
    adr2:
    adr3:
    adr4:
    adr5:
    adr6:
    adr7:
    adr8:
    adr9:
    adr10:
    adr11:
    adr12:
    adr13:
    adr14:
    adr15:
```

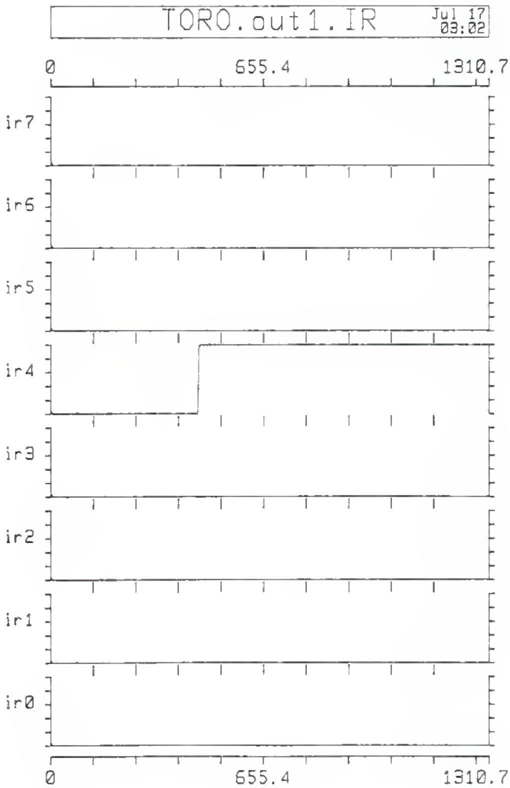

Figure B46: Plot Of Results From TORO.SIM1 Output From Instruction Register

Propagation Delays For The TORO.SIM1 Simulation
Output From The Instruction Register

```
Running SIMPLOT 1.3
                10%              50%              90%
file  TORO.out1.ir :
      ir0:
      ir1:
      ir2:
      ir3:
      ir4:
        454.00  1.55 v
        456.00                   4.79 v           4.79 v
      ir5:
      ir6:
      ir7:
```

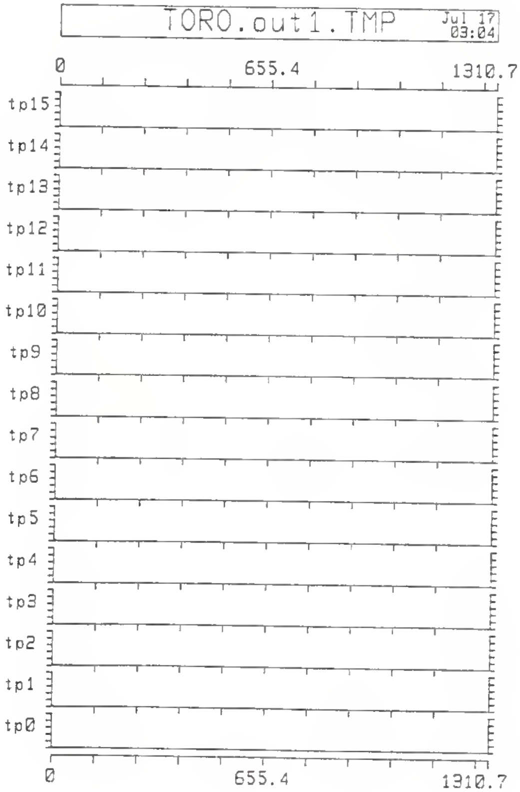

Figure B47: Plot Of Results From TORO.SIM1 Output From Temporary Register

Propagation Delays From TORO.SIM1 Simulation
Output From Temporary Register

```
Running SIMPLOT 1.3
                 10%                    50%                    90%
file  TORO.outl.tp :
        tp0:
        tp1:
        tp2:
        tp3:
        tp4:
        tp5:
        tp6:
        tp7:
        tp8:
        tp9:
       tp10:
       tp11:
       tp12:
       tp13:
       tp14:
       tp15:
```

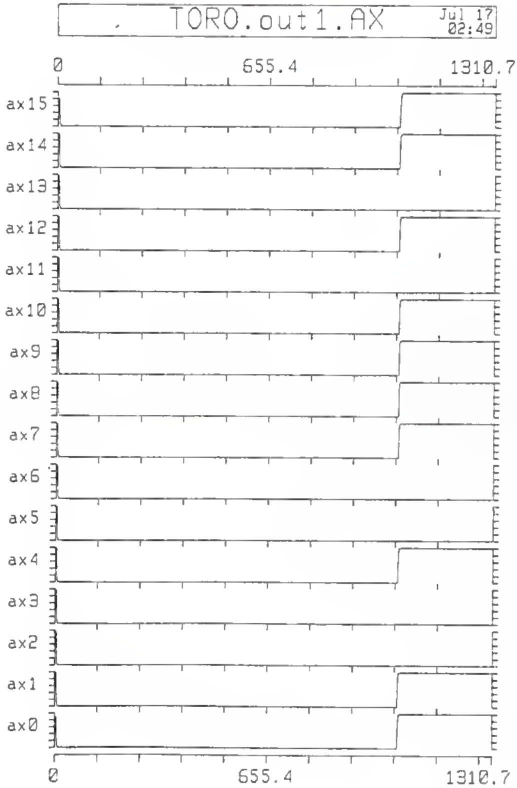

Figure B48: Plot Of Results From TORO.SIM1
Output From A/X Register Multiplexer

Propagation Delays For TORO.SIM1 Simulation
Output From Accumulator/Index Register Multiplexer

Running SIMPLOT 1.3

file TORO.outl.ax :		10%	50%	90%
ax0:				
	2.00	2.99 v	2.99 v	
	4.00			4.57 v
	6.00		0.94 v	0.94 v
	8.00	0.06 v		
	1058.00	3.28 v	3.28 v	
	1060.00			4.69 v
ax1:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
	1058.00	4.31 v	4.31 v	
	1060.00			4.97 v
ax2:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
ax3:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
ax4:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
	1058.00	4.31 v	4.31 v	
	1060.00			4.97 v
ax5:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
ax6:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
ax7:				
	2.00	4.09 v	4.09 v	
	4.00			4.92 v
	6.00	0.31 v	0.31 v	0.31 v
	1058.00	4.31 v	4.31 v	
	1060.00			4.97 v

ax8:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
1058.00	4.31 v	4.31 v	
1060.00			4.97 v
ax9:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
1058.00	4.31 v	4.31 v	
1060.00			4.97 v
ax10:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
1058.00	4.31 v	4.31 v	
1060.00			4.97 v
ax11:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
ax12:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
1058.00	4.31 v	4.31 v	
1060.00			4.97 v
ax13:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
ax14:			
2.00	4.09 v	4.09 v	
4.00			4.92 v
6.00	0.31 v	0.31 v	0.31 v
1058.00	4.31 v	4.31 v	
1060.00			4.97 v
ax15:			
2.00	2.37 v		
4.00		4.10 v	
6.00		1.24 v	
8.00	0.16 v		
1058.00	2.64 v	2.64 v	
1062.00			4.83 v

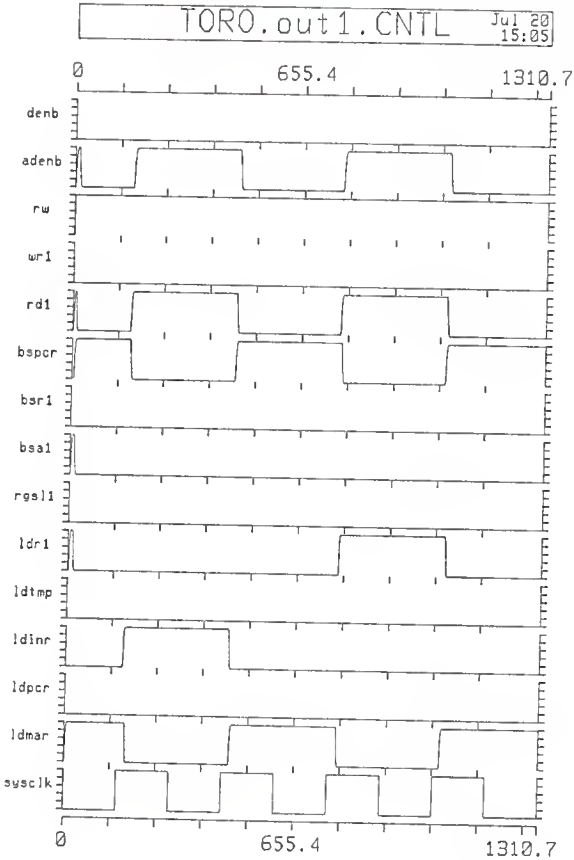

Figure B49: Plot Of Results From TORO.SIM1
Output From TORO Control Logic

Propagation Delays For TORO.SIM1 Simulation
Output From TORO Control Logic

Running SIMPLOT 1.3

	10%	50%	90%
file TORO.outl.cntl :			
sysclk:			
150.00	5.00 v	5.00 v	5.00 v
300.00	0.00 v	0.00 v	0.00 v
450.00	5.00 v	5.00 v	5.00 v
600.00	0.00 v	0.00 v	0.00 v
750.00	5.00 v	5.00 v	5.00 v
900.00	0.00 v	0.00 v	0.00 v
1050.00	5.00 v	5.00 v	5.00 v
1200.00	0.00 v	0.00 v	0.00 v
ldmar:			
2.00	0.59 v		
4.00		3.87 v	
6.00			4.84 v
174.00		0.83 v	0.83 v
176.00	0.04 v		
472.00	3.32 v	3.32 v	
474.00			4.75 v
776.00		0.92 v	0.92 v
778.00	0.04 v		
1070.00	3.48 v	3.48 v	
1072.00			4.78 v
ldpcr:			
ldinr:			
166.00	3.96 v	3.96 v	
168.00			4.89 v
466.00			4.40 v
468.00	0.32 v	0.32 v	
ldtmp:			
ldrl:			
4.00	1.86 v		
6.00		4.75 v	4.75 v
16.00		0.55 v	0.55 v
18.00	0.01 v		
774.00	4.38 v	4.38 v	
776.00			4.97 v
1076.00			4.34 v
1078.00	0.13 v	0.13 v	
rgsll:			

bsal:				
	4.00	3.45 v	3.45 v	
	6.00			4.74 v
	14.00		1.83 v	1.83 v
	16.00	0.12 v		
bsrl:				
bspcr:				
	8.00	2.83 v	2.83 v	
	12.00			4.88 v
	170.00		2.15 v	2.15 v
	172.00	0.26 v		
	470.00	3.15 v	3.15 v	
	472.00			4.54 v
	772.00		2.28 v	2.28 v
	774.00	0.28 v		
	1068.00	3.27 v	3.27 v	
	1070.00			4.57 v
rdl:				
	4.00	2.78 v	2.78 v	
	8.00			4.89 v
	12.00		1.20 v	1.20 v
	14.00	0.11 v		
	168.00	1.33 v		
	170.00		3.91 v	
	172.00			4.77 v
	472.00		2.49 v	2.49 v
	474.00	0.28 v		
	766.00	0.61 v		
	768.00		3.50 v	
	770.00			4.67 v
	1070.00			3.18 v
	1072.00	0.40 v	0.40 v	
wrl:				
rw:				
	2.00	4.92 v	4.92 v	4.92 v

adenb:

2.00	1.49 v		
4.00		3.49 v	
8.00			4.84 v
14.00			2.84 v
16.00		0.65 v	
18.00	0.11 v		
170.00	0.99 v		
172.00		3.16 v	
176.00			4.80 v
474.00			4.21 v
476.00		1.24 v	
478.00	0.23 v		
770.00	2.74 v	2.74 v	
774.00			4.73 v
1074.00		1.57 v	1.57 v
1076.00	0.30 v		

denb:

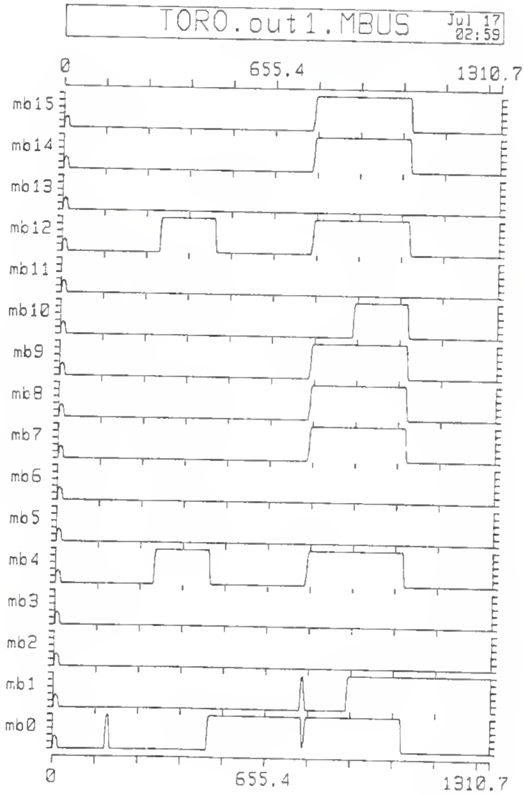

Figure B50: Plot Of Results From TORO.SIM1 Output From Main Internal Bus

Propagation Delay For The TORO.SIM1 Simulation
Output From Main Internal Bus

Running SIMPLOT 1.3

file TORO.out1.mb :

		10%	50%	90%
mb0:				
	2.00	1.65 v		
	16.00	0.30 v		
	160.00	0.54 v		
	162.00		2.76 v	
	166.00			4.86 v
	172.00			4.01 v
	174.00		1.35 v	
	176.00	0.16 v		
	476.00	2.69 v	2.69 v	
	480.00			4.85 v
	766.00			4.10 v
	768.00		1.38 v	
	776.00		2.87 v	
	780.00			4.87 v
	1072.00			4.25 v
	1074.00		1.31 v	
	1076.00	0.16 v		
mb1:				
	2.00	1.65 v		
	16.00	0.48 v		
	760.00	0.52 v		
	762.00		2.71 v	
	766.00			4.85 v
	770.00			4.36 v
	774.00		1.93 v	
	778.00	0.05 v		
	904.00	2.26 v		
	906.00		4.16 v	
	908.00			4.80 v
mb2:				
	2.00	1.53 v		
	18.00	0.06 v		
mb3:				
	2.00	1.61 v		
	16.00	0.49 v		

mb4:			
2.00	1.60 v		
16.00	0.49 v		
304.00	2.20 v		
306.00		4.10 v	
308.00			4.78 v
474.00			4.36 v
476.00		1.51 v	
478.00	0.21 v		
772.00	1.09 v		
776.00		2.80 v	
780.00			4.85 v
1072.00			4.30 v
1074.00		1.43 v	
1076.00	0.19 v		
mb5:			
2.00	1.59 v		
16.00	0.50 v		
mb6:			
2.00	1.57 v		
18.00	0.06 v		
mb7:			
2.00	1.56 v		
18.00	0.06 v		
772.00	1.08 v		
776.00		2.77 v	
780.00			4.84 v
1072.00			4.31 v
1074.00		1.49 v	
1076.00	0.21 v		
mb8:			
2.00	1.56 v		
18.00	0.06 v		
772.00	1.08 v		
776.00		2.79 v	
780.00			4.84 v
1072.00			4.29 v
1074.00		1.47 v	
1076.00	0.21 v		
mb9:			
2.00	1.54 v		
18.00	0.06 v		
772.00	1.07 v		
776.00		2.78 v	
780.00			4.83 v
1072.00			4.29 v
1074.00		1.49 v	
1076.00	0.22 v		

mb10:				
	2.00	1.53 v		
	18.00	0.06 v		
	904.00	2.11 v		
	906.00		4.00 v	
	908.00			4.74 v
	1072.00			4.30 v
	1074.00		1.51 v	
	1076.00	0.23 v		
mb11:				
	2.00	1.51 v		
	18.00	0.07 v		
mb12:				
	2.00	1.51 v		
	18.00	0.07 v		
	304.00	2.07 v		
	306.00		3.97 v	
	308.00			4.72 v
	474.00			4.40 v
	476.00		1.67 v	
	478.00	0.27 v		
	772.00	1.06 v		
	776.00		2.72 v	
	780.00			4.82 v
	1072.00			4.33 v
	1074.00		1.59 v	
	1076.00	0.25 v		
mb13:				
	2.00	1.49 v		
	18.00	0.07 v		
mb14:				
	2.00	1.48 v		
	18.00	0.07 v		
	772.00	1.06 v		
	776.00		2.70 v	
	780.00			4.81 v
	1072.00			4.34 v
	1074.00		1.62 v	
	1076.00	0.26 v		
mb15:				
	2.00	1.35 v		
	18.00	0.08 v		
	772.00	1.06 v		
	776.00		2.69 v	
	780.00			4.80 v
	1072.00			4.35 v
	1074.00		1.64 v	
	1076.00	0.27 v		

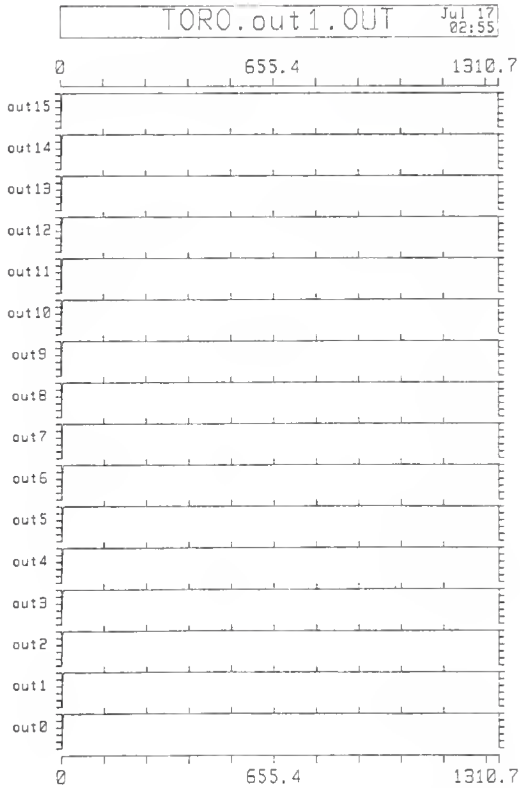

Figure B51: Plot Of Results From TORO.SIM1
Output From Write Register

Propagation Delays For The TORO.SIM1 Simulation
Output From Write Register

```
Running SIMPLOT 1.3
                10%                50%                 90%
file  TORO.outl.out :
    out0:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out1:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out2:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out3:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out4:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out5:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out6:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out7:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out8:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
    out9:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
   out10:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
   out11:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
   out12:
           2.00   2.84 v           2.84 v
           4.00                                        4.75 v
```

outl3:				
	2.00	2.84 v	2.84 v	
	4.00			4.75 v
outl4:				
	2.00	2.84 v	2.84 v	
	4.00			4.75 v
outl5:				
	2.00	2.84 v	2.84 v	
	4.00			4.75 v

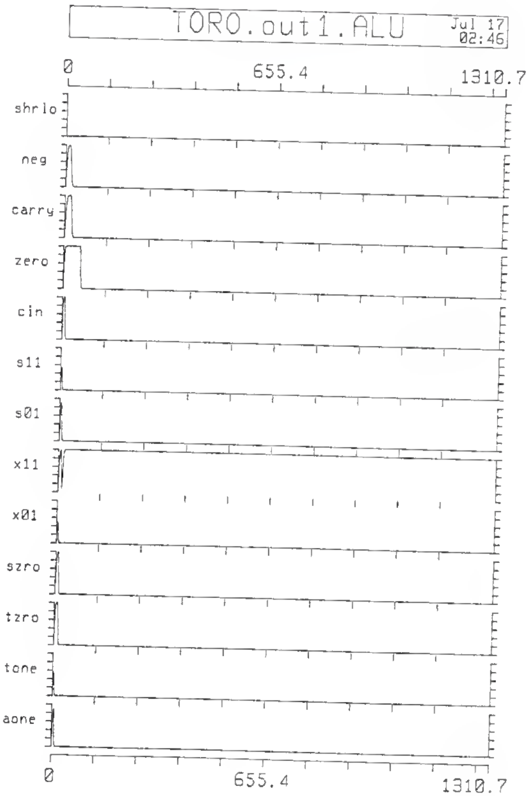

Figure B52: Plot Of Results From TORO.SIM1 Output From ALU Control Logic

Propagation Delays For The TORO.SIM1 Simulation
Output From ALU Control Logic

Running SIMPLOT 1.3

		10%	50%	90%
file TORO.outl.alu :				
aone:				
	2.00	2.03 v		
	4.00		4.28 v	
	8.00		0.77 v	
	10.00	0.06 v		
tone:				
	2.00	2.88 v	2.88 v	
	6.00	0.22 v	0.22 v	
tzro:				
	2.00	3.02 v	3.02 v	
	4.00			4.70 v
	12.00			2.77 v
	14.00	0.17 v	0.17 v	
szro:				
	2.00	2.52 v	2.52 v	
	6.00			4.91 v
	10.00		2.32 v	2.32 v
	12.00	0.20 v		
x01:				
	2.00	2.28 v		
	6.00	0.34 v		
x11:				
	2.00	2.31 v		
	4.00		4.20 v	
	6.00			4.81 v
	10.00			2.59 v
	12.00	0.39 v	0.39 v	
	14.00	2.18 v		
	16.00		4.14 v	
	18.00			4.79 v
s01:				
	2.00	2.79 v	2.79 v	
	4.00			4.55 v
	6.00			4.31 v
	8.00		0.63 v	
	10.00	0.04 v		
s11:				
	2.00	2.65 v	2.65 v	
	6.00	0.27 v	0.27 v	

cin:			
2.00	3.48 v	3.48 v	
4.00			4.86 v
10.00		0.82 v	0.82 v
12.00	0.02 v		
zero:			
2.00	3.02 v	3.02 v	
4.00			4.51 v
54.00		1.51 v	1.51 v
56.00	0.17 v		
carry:			
2.00	1.57 v		
4.00		3.19 v	
8.00			4.71 v
22.00		2.44 v	2.44 v
26.00	0.17 v		
neg:			
2.00	2.20 v		
4.00		3.80 v	
6.00			4.57 v
16.00			3.39 v
18.00		1.76 v	
22.00	0.15 v		
shrlo:			

TORO.SIM2: For this simulation, the TORO was allowed to load immediately a data word into the index register, then store the contents of the accumulator using indexed addressing. The data chosen for the indexed address causes the ALU to compute a zero output. In this way, a relative measure of the ALU delay was computed and used in calculating the maximum operating frequency for the TORO.

```
*******************************************************
*                                                     *
*                    TORO.SIM2                        *
*                                                     *
*      Simulation file for the LODX instruction       *
*       during immediate addressing, followed         *
*          by the store indexed instruction.          *
*                                                     *
*******************************************************
*
*  2/14/89 - Happy Valentine's Day!
*  2/16/89 - Updated after discovering
*            error in register enabling.
*  3/6/89  - Updating after discovering
*            unconnected ir bus line and
*            error in instruction generated
*            in this simulation file.
*  5/23/89 - Last set of simulations with
*            loads attached.
*  7/4/89  - Happy Birthday, America!
*            Final modifications
*
*******************************************************
*                                                     *
*  System Clock is set for 300 ns/cycle,              *
*  or 3.33 mHz.                                       *
*                                                     *
*******************************************************
*
*                      --T1--T2--T3--T0--T1--T2--T3--T4--T5--T0
cl clock  3000ns 0011001100110011001100110011001100110011
as clock sysclk
cl clkbar 3000ns 1100110011001100110011001100110011001100
as clkbar sysclkbar
cl rset   3000ns 0111111111111111111111111111111111111111
as rset sysrs
*
*
*******************************************************
*                                                     *
*  Plot the address outputs:                          *
*                                                     *
*******************************************************
*
pl adr0 adr1 adr2 adr3 adr4 adr5 adr6 adr7
pl adr8 adr9 adr10 adr11 adr12 adr13 adr14 adr15
*
```

```
**********************************************************
*                                                        *
*  Address loads added here:                             *
*                                                        *
**********************************************************
*
cap adr0 1.073pf
cap adr1 1.073pf
cap adr2 1.073pf
cap adr3 1.073pf
cap adr4 1.073pf
cap adr5 1.073pf
cap adr6 1.073pf
cap adr7 1.073pf
cap adr8 1.073pf
cap adr9 1.073pf
cap adr10 1.073pf
cap adr11 1.073pf
cap adr12 1.073pf
cap adr13 1.073pf
cap adr14 1.073pf
cap adr15 1.073pf
*
**********************************************************
*                                                        *
*  Plotting the output of the Instruction                *
*  Register, AX bus, TMP bus, and the mainbus:           *
*                                                        *
**********************************************************
*
pl ir0 ir1 ir2 ir3 ir4 ir5 ir6 ir7
pl ax0 ax1 ax2 ax3 ax4 ax5 ax6 ax7
pl ax8 ax9 ax10 ax11 ax12 ax13 ax14 ax15
pl tp0 tp1 tp2 tp3 tp4 tp5 tp6 tp7
pl tp8 tp9 tp10 tp11 tp12 tp13 tp14 tp15
pl mb0 mb1 mb2 mb3 mb4 mb5 mb6 mb7
pl mb8 mb9 mb10 mb11 mb12 mb13 mb14 mb15
*
**********************************************************
*                                                        *
*  Plotting the output to the data pads:                 *
*                                                        *
**********************************************************
*
pl out0 out1 out2 out3 out4 out5 out6 out7
pl out8 out9 out10 out11 out12 out13 out14 out15
*
```

```
**********************************************************
*                                                        *
*  Attaching the load for the data output here:          *
*                                                        *
**********************************************************
*
cap out0 0.665pf
cap out1 0.665pf
cap out2 0.665pf
cap out3 0.665pf
cap out4 0.665pf
cap out5 0.665pf
cap out6 0.665pf
cap out7 0.665pf
cap out8 0.665pf
cap out9 0.665pf
cap out10 0.665pf
cap out11 0.665pf
cap out12 0.665pf
cap out13 0.665pf
cap out14 0.665pf
cap out15 0.665pf
*
**********************************************************
*                                                        *
*  Plotting the TORO control signals:                    *
*                                                        *
**********************************************************
*
pl sysclk
pl ldmar ldpcr ldinr ldtmp ldrl rgsll
pl bsal bsrl bspcr rdl wrl
*
**********************************************************
*                                                        *
*  Plotting the pad enable and read/write                *
*  signals:                                              *
*                                                        *
**********************************************************
*
pl rw adenb denb
*
```

```
*********************************************************
*                                                       *
*  And, adding some capacitance. It turns               *
*  out that the enable input to the pads                *
*  is quite significant:                                *
*                                                       *
*********************************************************
*
cap rw 0.478pf
cap adenb 5.274pf
cap denb 4.834pf
*
*********************************************************
*                                                       *
*  Plotting the alu control signals:                    *
*                                                       *
*********************************************************
*
pl aone tone tzro szro x01 x11 s01 s11
pl cin zero carry neg shrlo
*
*********************************************************
*                                                       *
*  The instruction fetched and the data                 *
*  loaded:                                              *
*                                                       *
*********************************************************
*
*                   --T1--T2--T3--T0--T1--T2--T3--T4--T5--T0
*
cl in15 3000ns      0000000011111111000011111111000000000000
as in15 in15
cl in14 3000ns      0000000011111111111100000000000000000000
as in14 in14
cl in13 3000ns      0000000000000000111100001111111100000000
as in13 in13
cl in12 3000ns      0000111111111111000000000000000000000000
as in12 in12
*
*                   --T1--T2--T3--T0--T1--T2--T3--T4--T5--T0
*
cl in11 3000ns      0000000000000000111111110000111100000000
as in11 in11
cl in10 3000ns      0000000000001111111111110000000000000000
as in10 in10
cl in9  3000ns      0000000011111111000000000000000000000000
as in9 in9
cl in8 3000ns       0000000011111111000011110000000000000000
as in8 in8
```

```
*
*                  --T1--T2--T3--T0--T1--T2--T3--T4--T5--T0
*
cl in7 3000ns      0000000011111111000000000000000000000000
as in7 in7
cl in6 3000ns      0000000000000000000011110000111100000000
as in6 in6
cl in5 3000ns      0000000000000000000011110000111100000000
as in5 in5
cl in4 3000ns      0000111111111111111111110000000000000000
as in4 in4
*                  --T1--T2--T3--T0--T1--T2--T3--T4--T5--T0
*
cl in3 3000ns      0000000000000000000000000000111100000000
as in3 in3
cl in2 3000ns      0000000000000000000000000000111100000000
as in2 in2
cl in1 3000ns      0000000000001111111100000000000000000000
as in1 in1
cl in0 3000ns      0000111111111111111100000000111100000000
as in0 in0
*
*********************************************************
*                                                       *
*  The simulation parameters:                           *
*                                                       *
*  Plot Step:                       ps 5ns              *
*  Power Output? ( y = yes ):       po y                *
*  Simulation length:               sl                  *
*                                                       *
*********************************************************
*
cm + hpr
pf TORO.ClMi.CsMx
ti TORO.ClMi.CsMx
po y
sl 3000ns
ps 5ns
*
*********************************************************
*                                                       *
*  Power and Current computed after                     *
*  simulation by FACTS:                                 *
*                                                       *
*  Average Power:  13.0903 milliwatts                   *
*  Average Current:  2.61805 milliamps                  *
*                                                       *
*********************************************************
```

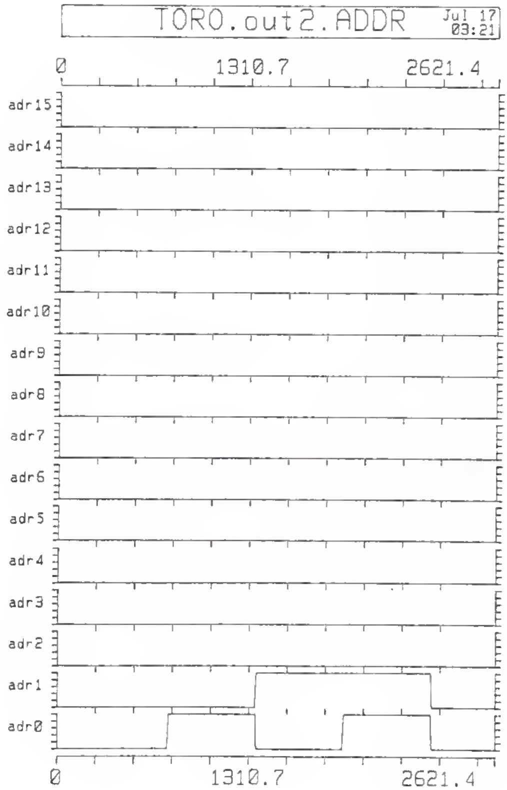

Figure B53: Plot Of Results From TORO.SIM2
Output From Memory Address Register

Propagation Delays For TORO.SIM2 Simulation
Output From Memory Address Register

```
Running SIMPLOT 1.3
                10%                   50%                 90%
file  TORO.out2.addr :
    adr0:
        755.00  2.15 v
        760.00                        4.90 v              4.90 v
        1360.00 0.42 v                0.42 v              0.42 v
        1955.00 2.15 v
        1960.00                       4.90 v              4.90 v
        2560.00 0.42 v                0.42 v              0.42 v
    adr1:
        1355.00 2.15 v
        1360.00                       4.90 v              4.90 v
        2560.00 0.42 v                0.42 v              0.42 v
    adr2:
    adr3:
    adr4:
    adr5:
    adr6:
    adr7:
    adr8:
    adr9:
   adr10:
   adr11:
   adr12:
   adr13:
   adr14:
   adr15:
```

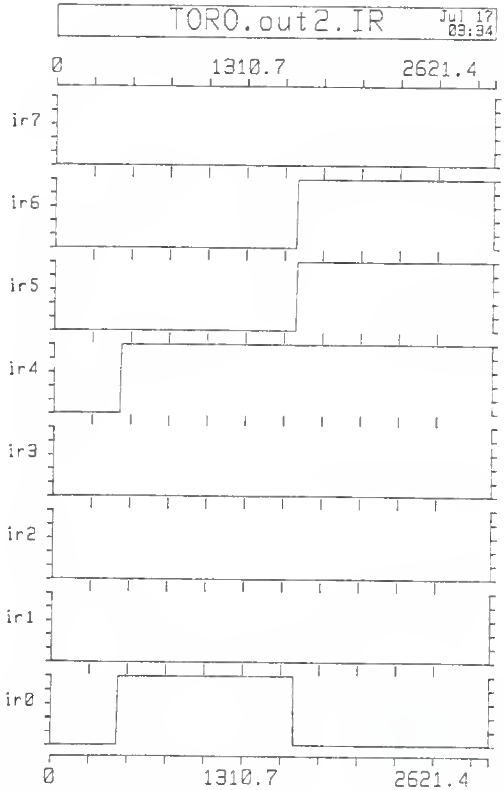

Figure B54: Plot Of Results From TORO.SIM2
Output Of Instruction Register

Propagation Delays For The TORO.SIM2 Simulation
Output Of Instruction Register

```
Running SIMPLOT 1.3
                10%              50%              90%
file  TORO.out2.ir :
     ir0:
        455.00  3.12 v           3.12 v
        460.00                                    4.99 v
        1660.00 0.07 v           0.07 v           0.07 v
     ir1:
     ir2:
     ir3:
     ir4:
        455.00  2.81 v           2.81 v
        460.00                                    4.98 v
     ir5:
        1655.00 2.85 v           2.85 v
        1660.00                                   4.98 v
     ir6:
        1655.00 2.69 v           2.69 v
        1660.00                                   4.97 v
     ir7:
```

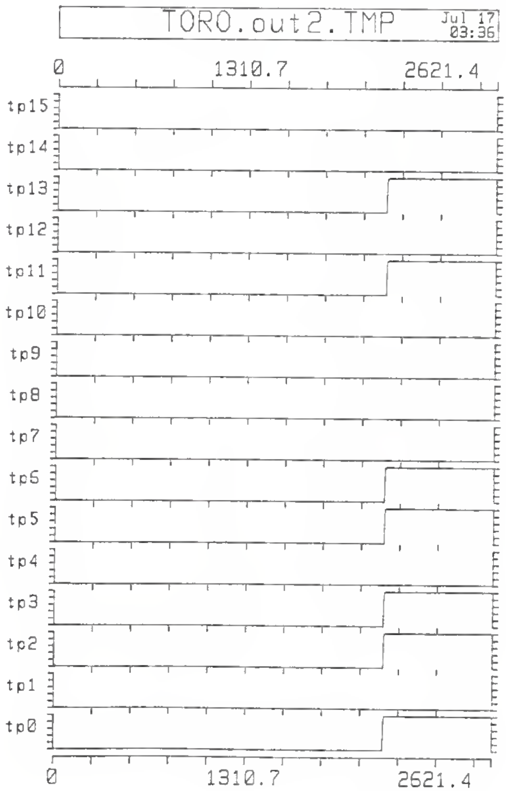

Figure B55: Plot Of Results From TORO.SIM2 Output From Temporary Register

Propagation Delays For TORO.SIM2 Simulation
Output From Temporary Register

```
Running SIMPLOT 1.3
                  10%                   50%                 90%
file  TORO.out2.tp :
      tp0:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
      tp1:
      tp2:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
      tp3:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
      tp4:
      tp5:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
      tp6:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
      tp7:
      tp8:
      tp9:
     tp10:
     tp11:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
     tp12:
     tp13:
         2255.00 4.39 v                4.39 v
         2260.00                                           4.99 v
     tp14:
     tp15:
```

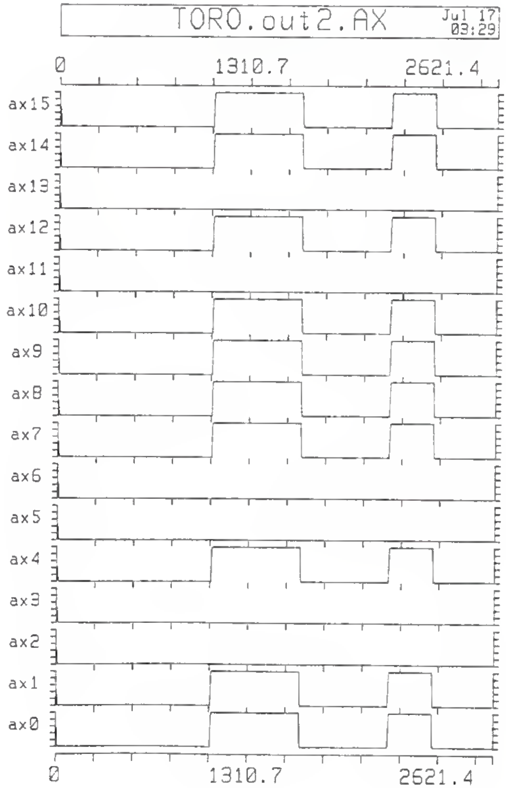

Figure B56: Plot Of Results From TORO.SIM2
Output From A/X Register Multiplexer

Propagation Delays For TORO.SIM2 Simulation
Output From Accumulator/Index Register Multiplexer

Running SIMPLOT 1.3

file TORO.out2.ax :

		10%	50%	90%
ax0:				
	5.00	2.87 v	2.87 v	
	10.00	0.01 v	0.01 v	
	1060.00	4.20 v	4.20 v	
	1065.00			4.99 v
	1670.00	0.23 v	0.23 v	0.23 v
	2275.00	1.50 v		
	2280.00		4.95 v	4.95 v
	2580.00	0.13 v	0.13 v	0.13 v
ax1:				
	5.00	2.26 v		
	10.00	0.01 v		
	1060.00	4.83 v	4.83 v	4.83 v
	1670.00	0.02 v	0.02 v	0.02 v
	2275.00	2.33 v		
	2280.00		4.99 v	4.99 v
	2580.00	0.01 v	0.01 v	0.01 v
ax2:				
	5.00	2.26 v		
	10.00	0.01 v		
ax3:				
	5.00	2.26 v		
	10.00	0.01 v		
ax4:				
	5.00	2.26 v		
	10.00	0.01 v		
	1060.00	4.83 v	4.83 v	4.83 v
	1670.00	0.02 v	0.02 v	0.02 v
	2275.00	2.33 v		
	2280.00		4.99 v	4.99 v
	2580.00	0.01 v	0.01 v	0.01 v
ax5:				
	5.00	2.26 v		
	10.00	0.01 v		
ax6:				
	5.00	2.26 v		
	10.00	0.01 v		

```
ax7:
      5.00  2.26 v
     10.00  0.01 v
   1060.00 4.83 v          4.83 v          4.83 v
   1670.00 0.02 v          0.02 v          0.02 v
   2275.00 2.33 v
   2280.00                 4.99 v          4.99 v
   2580.00 0.01 v          0.01 v          0.01 v
ax8:
      5.00  2.26 v
     10.00  0.01 v
   1060.00 4.83 v          4.83 v          4.83 v
   1670.00 0.02 v          0.02 v          0.02 v
   2275.00 2.33 v
   2280.00                 4.99 v          4.99 v
   2580.00 0.01 v          0.01 v          0.01 v
ax9:
      5.00  2.26 v
     10.00  0.01 v
   1060.00 4.83 v          4.83 v          4.83 v
   1670.00 0.02 v          0.02 v          0.02 v
   2275.00 2.33 v
   2280.00                 4.99 v          4.99 v
   2580.00 0.01 v          0.01 v          0.01 v
ax10:
      5.00  2.26 v
     10.00  0.01 v
   1060.00 4.83 v          4.83 v          4.83 v
   1670.00 0.02 v          0.02 v          0.02 v
   2275.00 2.33 v
   2280.00                 4.99 v          4.99 v
   2580.00 0.01 v          0.01 v          0.01 v
ax11:
      5.00  2.26 v
     10.00  0.01 v
ax12:
      5.00  2.26 v
     10.00  0.01 v
   1060.00 4.83 v          4.83 v          4.83 v
   1670.00 0.02 v          0.02 v          0.02 v
   2275.00 2.33 v
   2280.00                 4.99 v          4.99 v
   2580.00 0.01 v          0.01 v          0.01 v
ax13:
      5.00  2.26 v
     10.00  0.01 v
```

axl4:			
5.00	2.26 v		
10.00	0.01 v		
1060.00	4.83 v	4.83 v	4.83 v
1670.00	0.02 v	0.02 v	0.02 v
2275.00	2.33 v		
2280.00		4.99 v	4.99 v
2580.00	0.01 v	0.01 v	0.01 v
axl5:			
5.00	2.83 v	2.83 v	
10.00	0.02 v	0.02 v	
1060.00	3.61 v	3.61 v	
1065.00			4.96 v
1670.00		0.58 v	0.58 v
1675.00	0.01 v		
2275.00	1.15 v		
2280.00		4.82 v	4.82 v
2580.00	0.37 v	0.37 v	0.37 v

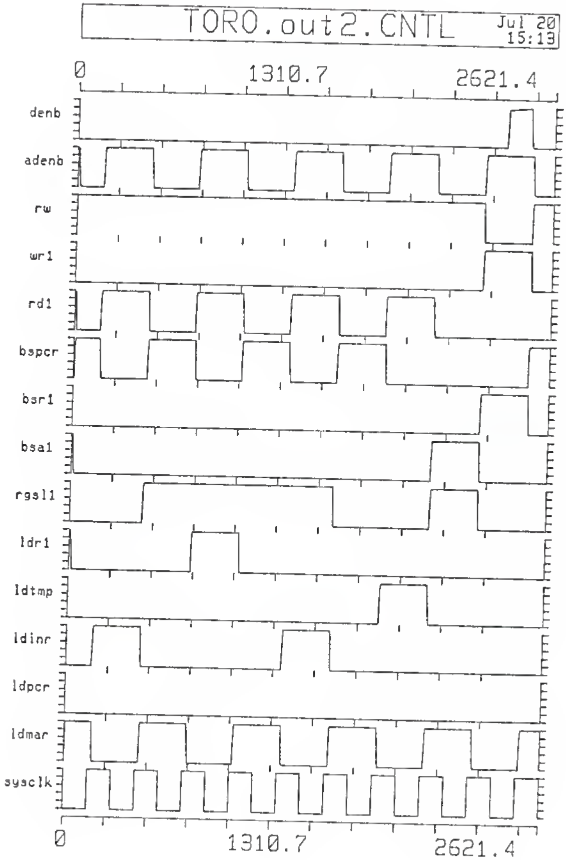

Figure B57: Plot Of Results From TORO.SIM2 Output From TORO Control Logic

Propagation Delays For The TORO.SIM2 Simulation
Output From TORO Control Logic

Running SIMPLOT 1.3

file TORO.out2.cntl :

		10%	50%	90%
sysclk:				
	150.00	5.00 v	5.00 v	5.00 v
	300.00	0.00 v	0.00 v	0.00 v
	450.00	5.00 v	5.00 v	5.00 v
	600.00	0.00 v	0.00 v	0.00 v
	750.00	5.00 v	5.00 v	5.00 v
	900.00	0.00 v	0.00 v	0.00 v
	1050.00	5.00 v	5.00 v	5.00 v
	1200.00	0.00 v	0.00 v	0.00 v
	1350.00	5.00 v	5.00 v	5.00 v
	1500.00	0.00 v	0.00 v	0.00 v
	1650.00	5.00 v	5.00 v	5.00 v
	1800.00	0.00 v	0.00 v	0.00 v
	1950.00	5.00 v	5.00 v	5.00 v
	2100.00	0.00 v	0.00 v	0.00 v
	2250.00	5.00 v	5.00 v	5.00 v
	2400.00	0.00 v	0.00 v	0.00 v
	2550.00	5.00 v	5.00 v	5.00 v
	2700.00	0.00 v	0.00 v	0.00 v
	2850.00	5.00 v	5.00 v	5.00 v
ldmar:				
	5.00	4.56 v	4.56 v	4.56 v
	175.00	0.19 v	0.19 v	0.19 v
	475.00	4.91 v	4.91 v	4.91 v
	775.00			3.03 v
	780.00	0.01 v	0.01 v	
	1070.00	3.48 v	3.48 v	
	1075.00			4.99 v
	1375.00	0.19 v	0.19 v	0.19 v
	1675.00	4.91 v	4.91 v	4.91 v
	1975.00			3.03 v
	1980.00	0.01 v	0.01 v	
	2270.00	2.15 v		
	2275.00		4.98 v	4.98 v
	2575.00		0.78 v	0.78 v
	2580.00	0.01 v		
	2870.00	3.93 v	3.93 v	
	2875.00			4.99 v

ldpcr:			
ldinr:			
165.00	2.34 v		
170.00		4.99 v	4.99 v
470.00	0.01 v	0.01 v	0.01 v
1365.00	2.34 v		
1370.00		4.99 v	4.99 v
1670.00	0.01 v	0.01 v	0.01 v
ldtmp:			
1970.00	4.30 v	4.30 v	
1975.00			4.99 v
2270.00		2.07 v	2.07 v
2275.00	0.01 v		
ldrl:			
5.00	3.99 v	3.99 v	
10.00			4.99 v
15.00			3.20 v
20.00	0.01 v	0.01 v	
775.00	4.86 v	4.86 v	4.86 v
1080.00	0.01 v	0.01 v	0.01 v
rgsll:			
460.00	4.17 v	4.17 v	
465.00			4.99 v
1665.00	0.02 v	0.02 v	0.02 v
2270.00	3.20 v	3.20 v	
2275.00			4.99 v
2575.00	0.01 v	0.01 v	0.01 v
bsal:			
5.00	4.34 v	4.34 v	
10.00			4.99 v
15.00		0.51 v	0.51 v
20.00	0.01 v		
2270.00	2.97 v	2.97 v	
2275.00			4.98 v
2575.00	0.03 v	0.03 v	0.03 v
bsrl:			
2575.00	4.97 v	4.97 v	4.97 v
2875.00	0.01 v	0.01 v	0.01 v

bspcr:

10.00	4.43 v	4.43 v	
15.00			4.99 v
170.00		2.15 v	2.15 v
175.00	0.01 v		
470.00	3.15 v	3.15 v	
475.00			4.96 v
775.00	0.09 v	0.09 v	0.09 v
1070.00	4.57 v	4.57 v	4.57 v
1370.00		2.14 v	2.14 v
1375.00	0.01 v		
1670.00	3.15 v	3.15 v	
1675.00			4.95 v
1975.00	0.09 v	0.09 v	0.09 v
2870.00	4.68 v	4.68 v	4.68 v

rdl:

5.00	3.86 v	3.86 v	
10.00			4.84 v
15.00	0.03 v	0.03 v	0.03 v
170.00	3.91 v	3.91 v	
175.00			4.98 v
475.00	0.08 v	0.08 v	0.08 v
770.00	4.67 v	4.67 v	4.67 v
1070.00			3.18 v
1075.00	0.01 v	0.01 v	
1370.00	3.91 v	3.91 v	
1375.00			4.98 v
1675.00	0.08 v	0.08 v	0.08 v
1970.00	4.67 v	4.67 v	4.67 v
2270.00		1.07 v	1.07 v
2275.00	0.01 v		

wrl:

2575.00	4.81 v	4.81 v	4.81 v
2875.00	0.07 v	0.07 v	0.07 v

rw:

5.00	4.99 v	4.99 v	4.99 v
2575.00	0.01 v	0.01 v	0.01 v
2875.00	4.99 v	4.99 v	4.99 v

adenb:

5.00	4.10 v	4.10 v	
10.00			4.95 v
15.00		1.45 v	1.45 v
20.00	0.02 v		
170.00	0.99 v		
175.00		4.63 v	4.63 v
475.00			2.51 v
480.00	0.04 v	0.04 v	
770.00	2.74 v	2.74 v	
775.00			4.85 v
1075.00		0.71 v	0.71 v
1080.00	0.01 v		
1370.00	0.99 v		
1375.00		4.63 v	4.63 v
1675.00			2.53 v
1680.00	0.04 v	0.04 v	
1970.00	2.74 v	2.74 v	
1975.00			4.85 v
2275.00	0.26 v	0.26 v	0.26 v
2575.00	3.65 v	3.65 v	
2580.00			4.92 v
2875.00		1.61 v	1.61 v
2880.00	0.02 v		

denb:

2705.00	1.51 v		
2710.00		4.78 v	4.78 v
2855.00		1.60 v	1.60 v
2860.00	0.01 v		

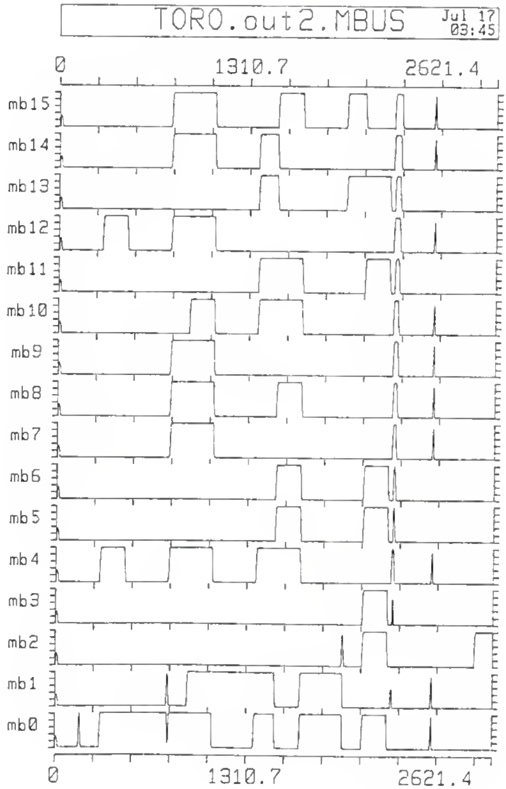

Figure B58: Plot Of Results From TORO.SIM2 Output From Main Internal Bus

Propagation Delays For The TORO.SIM2 Simulation
Output From Main Internal Bus

Running SIMPLOT 1.3

	10%	50%	90%
file TORO.out2.mb :			
mb0:			
5.00	1.69 v		
20.00	0.01 v		
160.00	0.54 v		
165.00		4.70 v	4.70 v
175.00	0.49 v	0.49 v	0.49 v
305.00	3.41 v	3.41 v	
310.00			4.96 v
770.00		0.52 v	0.52 v
780.00		4.87 v	4.87 v
1075.00	0.48 v	0.48 v	0.48 v
1360.00	0.54 v		
1365.00		4.70 v	4.70 v
1505.00		1.28 v	1.28 v
1510.00	0.01 v		
1675.00	1.44 v		
1680.00		4.85 v	4.85 v
1970.00	0.08 v	0.08 v	0.08 v
2105.00	3.41 v	3.41 v	
2110.00			4.96 v
2275.00		0.70 v	0.70 v
2280.00	0.01 v		
2575.00	1.93 v		
2580.00		4.75 v	4.75 v
2585.00	0.24 v	0.24 v	0.24 v

mb1:			
5.00	1.69 v		
20.00	0.01 v		
760.00	0.52 v		
765.00		4.68 v	4.68 v
770.00			4.36 v
775.00		1.31 v	
780.00	0.01 v		
905.00	3.39 v	3.39 v	
910.00			4.96 v
1505.00		1.30 v	1.30 v
1510.00	0.01 v		
1675.00	1.43 v		
1680.00		4.85 v	4.85 v
1970.00	0.07 v	0.07 v	0.07 v
2300.00	2.62 v	2.62 v	
2310.00	0.02 v	0.02 v	
2575.00	1.92 v		
2580.00		4.47 v	
2585.00	0.12 v	0.12 v	
mb2:			
5.00	1.58 v		
20.00	0.01 v		
1965.00	4.56 v	4.56 v	4.56 v
1970.00			4.39 v
1975.00		1.37 v	
1980.00	0.01 v		
2105.00	3.22 v	3.22 v	
2110.00			4.94 v
2275.00		0.86 v	0.86 v
2280.00	0.01 v		
2875.00	3.40 v	3.40 v	
2880.00			4.95 v
mb3:			
5.00	1.66 v		
20.00	0.01 v		
2105.00	3.35 v	3.35 v	
2110.00			4.95 v
2275.00		0.75 v	0.75 v
2280.00	0.01 v		
2310.00	3.73 v	3.73 v	
2315.00	0.11 v	0.11 v	

mb4:

5.00	1.65 v		
20.00	0.01 v		
305.00	3.32 v	3.32 v	
310.00			4.95 v
475.00			3.02 v
480.00	0.02 v	0.02 v	
775.00	1.65 v		
780.00		4.85 v	4.85 v
1075.00		0.56 v	0.56 v
1080.00	0.01 v		
1375.00	3.34 v	3.34 v	
1380.00			4.95 v
1675.00			3.02 v
1680.00	0.02 v	0.02 v	
2300.00	2.57 v	2.57 v	
2305.00			4.92 v
2315.00			4.37 v
2320.00	0.08 v	0.08 v	
2575.00	1.87 v		
2580.00		4.45 v	
2585.00	0.13 v	0.13 v	

mb5:

5.00	1.64 v		
20.00	0.01 v		
1505.00	3.30 v	3.30 v	
1510.00			4.95 v
1675.00			3.04 v
1680.00	0.02 v	0.02 v	
2105.00	3.30 v	3.30 v	
2110.00			4.95 v
2275.00		0.79 v	0.79 v
2280.00	0.01 v		
2310.00	4.44 v	4.44 v	
2315.00			4.99 v
2320.00		1.18 v	1.18 v
2325.00	0.01 v		

mb6:

5.00	1.62 v		
20.00	0.01 v		
1505.00	3.28 v	3.28 v	
1510.00			4.95 v
1675.00			3.05 v
1680.00	0.03 v	0.03 v	
2105.00	3.28 v	3.28 v	
2110.00			4.95 v
2275.00		0.80 v	0.80 v
2280.00	0.01 v		
2310.00	4.43 v	4.43 v	
2315.00			4.99 v
2320.00			4.06 v
2325.00	0.06 v	0.06 v	

mb7:

5.00	1.60 v		
20.00	0.01 v		
775.00	1.64 v		
780.00		4.84 v	4.84 v
1075.00		0.60 v	0.60 v
1080.00	0.01 v		
2300.00	2.51 v	2.51 v	
2305.00			4.91 v
2325.00		1.75 v	1.75 v
2330.00	0.01 v		
2575.00	1.83 v		
2580.00		4.43 v	
2585.00	0.14 v	0.14 v	

mb8:

5.00	1.60 v		
20.00	0.01 v		
775.00	1.65 v		
780.00		4.84 v	4.84 v
1075.00		0.59 v	0.59 v
1080.00	0.01 v		
1505.00	3.25 v	3.25 v	
1510.00			4.94 v
1675.00			3.03 v
1680.00	0.03 v	0.03 v	
2300.00	2.50 v		
2305.00		4.90 v	4.90 v
2330.00		1.08 v	1.08 v
2335.00	0.01 v		
2575.00	1.82 v		
2580.00		4.42 v	
2585.00	0.15 v	0.15 v	

mb9:			
5.00	1.59 v		
20.00	0.01 v		
775.00	1.65 v		
780.00		4.83 v	4.83 v
1075.00		0.60 v	0.60 v
1080.00	0.01 v		
2300.00	2.48 v		
2305.00		4.90 v	4.90 v
2335.00	0.17 v	0.17 v	0.17 v
2575.00	1.80 v		
2580.00		4.41 v	
2585.00	0.15 v	0.15 v	
mb10:			
5.00	1.57 v		
20.00	0.01 v		
905.00	3.21 v	3.21 v	
910.00			4.94 v
1075.00		0.62 v	0.62 v
1080.00	0.01 v		
1375.00	3.26 v	3.26 v	
1380.00			4.94 v
1675.00			3.06 v
1680.00	0.03 v	0.03 v	
2300.00	2.46 v		
2305.00		4.89 v	4.89 v
2335.00		0.93 v	0.93 v
2340.00	0.01 v		
2575.00	1.79 v		
2580.00		4.40 v	
2585.00	0.16 v	0.16 v	
mb11:			
5.00	1.56 v		
20.00	0.01 v		
1375.00	3.24 v	3.24 v	
1380.00			4.94 v
1675.00			3.08 v
1680.00	0.03 v	0.03 v	
2105.00	3.19 v	3.19 v	
2110.00			4.93 v
2275.00		0.88 v	0.88 v
2280.00	0.01 v		
2310.00	4.36 v	4.36 v	
2315.00			4.98 v
2335.00			4.27 v
2340.00	0.09 v	0.09 v	

mb12:

5.00	1.55 v		
20.00	0.01 v		
305.00	3.17 v	3.17 v	
310.00			4.93 v
475.00			3.13 v
480.00	0.03 v	0.03 v	
775.00	1.63 v		
780.00		4.82 v	4.82 v
1075.00		0.66 v	0.66 v
1080.00	0.01 v		
2300.00	2.42 v		
2305.00		4.89 v	4.89 v
2340.00			4.20 v
2345.00	0.09 v	0.09 v	
2575.00	1.76 v		
2580.00		4.38 v	
2585.00	0.17 v	0.17 v	

mb13:

5.00	1.54 v		
20.00	0.01 v		
1375.00	3.19 v	3.19 v	
1380.00			4.93 v
1505.00		1.54 v	1.54 v
1510.00	0.01 v		
1975.00	1.63 v		
1980.00		4.81 v	4.81 v
2275.00		0.92 v	0.92 v
2280.00	0.01 v		
2310.00	4.33 v	4.33 v	
2315.00			4.98 v
2345.00		1.38 v	1.38 v
2350.00	0.01 v		

mb14:

5.00	1.52 v		
20.00	0.01 v		
775.00	1.63 v		
780.00		4.81 v	4.81 v
1075.00		0.69 v	0.69 v
1080.00	0.01 v		
1375.00	3.18 v	3.18 v	
1380.00			4.93 v
1505.00		1.56 v	1.56 v
1510.00	0.01 v		
2300.00	2.39 v		
2305.00		4.88 v	4.88 v
2345.00			4.43 v
2350.00	0.12 v	0.12 v	
2575.00	1.74 v		
2580.00		4.36 v	
2585.00	0.17 v	0.17 v	

mb15:

5.00	1.38 v		
20.00	0.01 v		
775.00	1.63 v		
780.00		4.80 v	4.80 v
1075.00		0.71 v	0.71 v
1080.00	0.01 v		
1505.00	3.11 v	3.11 v	
1510.00			4.92 v
1675.00			3.18 v
1680.00	0.04 v	0.04 v	
1975.00	1.63 v		
1980.00		4.80 v	4.80 v
2105.00		1.58 v	1.58 v
2110.00	0.01 v		
2305.00	3.72 v	3.72 v	
2310.00			4.95 v
2355.00		0.53 v	0.53 v
2360.00	0.01 v		
2575.00	1.72 v		
2580.00		4.74 v	4.74 v
2585.00		0.59 v	0.59 v
2590.00	0.01 v		

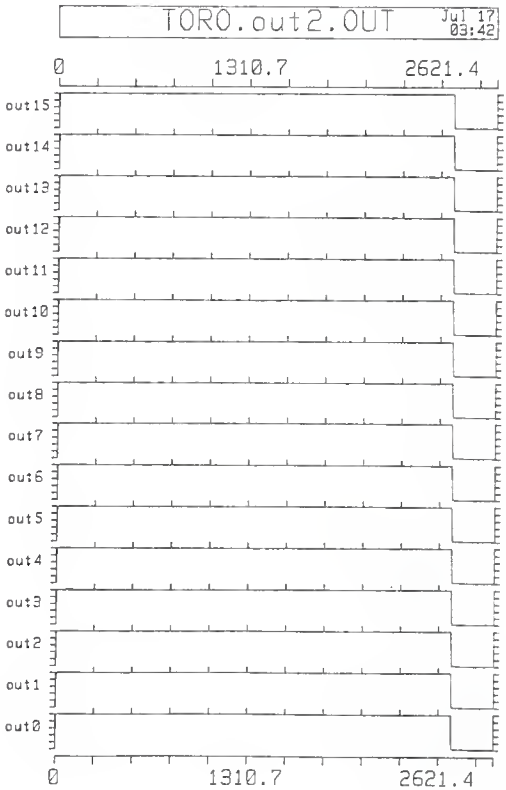

Figure B59: Plot Of Results From TORO.SIM2 Output From Write Register

Propagation Delays For The TORO.SIM2 Simulation
Output From Write Register

```
Running SIMPLOT 1.3
                10%                 50%             90%
file  TORO.out2.out :
    out0:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out1:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out2:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out3:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out4:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out5:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out6:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out7:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out8:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
    out9:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
   out10:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
   out11:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
   out12:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
   out13:
           5.00   4.92 v          4.92 v          4.92 v
        2710.00   0.07 v          0.07 v          0.07 v
```

```
outl4:
       5.00  4.92 v          4.92 v          4.92 v
    2710.00 0.07 v           0.07 v          0.07 v
outl5:
       5.00  4.92 v          4.92 v          4.92 v
    2710.00 0.07 v           0.07 v          0.07 v
```

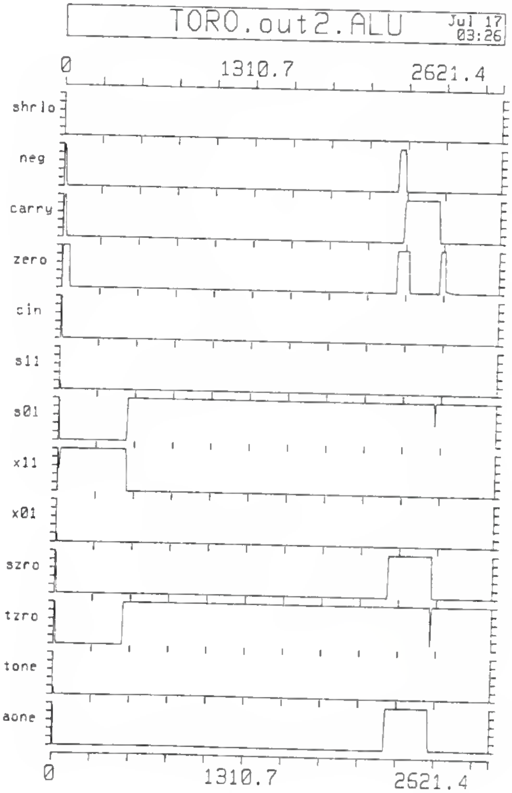

Figure B60: Plot Of Results From TORO.SIM1 Output From ALU Control Logic

Propagation Delays For The TORO.SIM2 Simulation
Output From ALU Control Logic

Running SIMPLOT 1.3

file TORO.out2.alu :

	10%	50%	90%
aone:			
5.00	4.69 v	4.69 v	4.69 v
10.00	0.06 v	0.06 v	0.06 v
2280.00	3.30 v	3.30 v	
2285.00			4.97 v
2580.00		0.99 v	0.99 v
2585.00	0.01 v		
tone:			
5.00	0.87 v		
10.00	0.01 v		
tzro:			
5.00	4.89 v	4.89 v	4.89 v
15.00	0.04 v	0.04 v	0.04 v
475.00	4.74 v	4.74 v	4.74 v
2580.00	0.34 v	0.34 v	0.34 v
2585.00	3.96 v	3.96 v	
2590.00			4.99 v
szro:			
5.00	4.79 v	4.79 v	4.79 v
10.00		2.32 v	2.32 v
15.00	0.01 v		
2285.00	4.89 v	4.89 v	4.89 v
2585.00	0.06 v	0.06 v	0.06 v
x01:			
5.00	1.00 v		
10.00	0.01 v		
x11:			
5.00	4.60 v	4.60 v	4.60 v
10.00			2.59 v
20.00			4.95 v
470.00			3.99 v
475.00	0.03 v	0.03 v	
s01:			
5.00	4.81 v	4.81 v	4.81 v
10.00	0.04 v	0.04 v	0.04 v
470.00	0.80 v		
475.00		4.92 v	4.92 v
2580.00		2.31 v	2.31 v
2585.00		4.92 v	4.92 v
s11:			
5.00	0.94 v		
10.00	0.01 v		

cin:			
5.00	4.96 v	4.96 v	4.96 v
10.00		0.82 v	0.82 v
15.00	0.01 v		
zero:			
5.00	4.77 v	4.77 v	4.77 v
55.00		0.54 v	0.54 v
60.00	0.01 v		
2300.00	4.33 v	4.33 v	
2305.00			4.99 v
2385.00		0.88 v	0.88 v
2390.00	0.01 v		
2595.00	3.46 v	3.46 v	
2600.00			4.97 v
2635.00	0.26 v	0.26 v	0.26 v
carry:			
5.00	3.79 v	3.79 v	
10.00			4.90 v
25.00	0.36 v	0.36 v	0.36 v
2340.00	0.86 v		
2345.00		3.48 v	
2350.00			4.83 v
2585.00			2.86 v
2590.00	0.10 v	0.10 v	
neg:			
5.00	4.27 v	4.27 v	
10.00			4.93 v
15.00			4.07 v
20.00		0.58 v	
25.00	0.02 v		
2295.00	0.57 v		
2300.00		3.96 v	
2305.00			4.92 v
2345.00			4.05 v
2350.00		0.56 v	
2355.00	0.01 v		
shrlo:			

9.0 Appendix C - TORO 680-16 Library

In this appendix appears the paper by Devore and Hardin used in constructing this microprocessor. That reprint is followed by an up-dated version of the register-transfer notation for the store instruction during direct and indexed addressing. Recall that the original TORO by Devore did not include a write data register.

This notation is then followed by the data input equations for the 4-bit counter macro used to construct the TORO 680-16 program counter. Another set of equations, for the phase clock T is also given. Finally, the set of equations used to generate the ALU control signals, and the read/write, address output pad enable and data output pad enable control signals are given.

A Computer Design for Introducting Hardware and Software Concepts

JOHN J. DEVORE, MEMBER, IEEE, AND DAVID S. HARDIN, STUDENT MEMBER, IEEE

Abstract—This paper describes a simple computer design and an associated full screen PC-based simulator used to teach basic hardware and software concepts in Introduction to Computer Engineering, a sophomore level course at Kansas State University. The culmination of the course is the presentation of a complete CPU which could be built from the SSI and MSI circuits studied in the course. The CPU design is simple enough to enable equations for a hardwired control unit to be displayed during simulation, yet powerful enough to introduce Assembly language programming using the same architecture.

Introduction to Computer Engineering is a course taught in the Department of Electrical and Computer Engineering that is required of all EECE and computer science students at Kansas State University. The annual enrollment is approximately 450 students. The course covers the usual topics of Boolean algebra, combinational design, sequential design, computer arithmetic, arithmetic and logic units, memory systems, and computer organization. Also included, however, is a case study of a simple computer and Assembly language programming. Most textbooks omit the case study [1], [2], study a derivative of the PDP-8 or PDP-11 [3], [4], or study one or more processors primarily from a programmer's point of view [5]-[7]. We wanted to present a computer that was simple enough to be developed from the hardware already presented in the course, yet powerful enough to introduce Assembly lanaguage programming. We, therefore, undertook the design of our own processor to satisfy the above goals.

One of the reasons for including Assembly language programming in the course is to lay a foundation for a microcomputer systems design course which utilizes Motorla 6800 and 68000 microprocessors; thus, an attempt was made to make the CPU similar to (a scaled-down version of) the 6800 [8]. The resulting machine was named the TORO 680 (scaled-down MOTOROLA 6800). It is a byte-oriented, single-operand machine, with 256 bytes of memory. Two registers are accessible to the programmer—an accumulator and an index register. The control unit is a hardwired design with 17 inputs and 14 outputs.

A tool to aid the student in understanding the operation of the processor is provided in a PC-based full-screen simulator. It dynamically displays the Boolean equations for the control unit as well as the user registers, internal registers, and memory.

Manuscript received August 31, 1987.

The authors are with the Department of Electrical and Computer Engineering, Kansas State University, Manhattan, KS 66506

IEEE Log Number 8717419.

Architecture

The architectural goals were shaped by the TORO's use as a "teaching machine," and include the following.

- It must have a simple yet powerful instruction set.
- The 8-bit opcode is to be encoded by fields.
- There should be minimal usage of field redefinition.
- A variety of addressing modes should be supported.
- The instruction set should be as regular (uniform use of addressing modes for all instructions) as possible.
- A representative set of ALU functions should be used.
- Minimal parallelism should be used within the hardware as sequential events are easier to comprehend.
- It should resemble a reduced Motorola 6800 microprocessor.

In the course of making the tradeoffs that are inevitably required during the design process, the following features were (reluctantly) dropped:

- A hardware stack pointer.
- Hardware subroutine call and return (software support can be effected).
- Programmed I/O (memory-mapped I/O is to be defined in a later version).
- All status flags except carry, negative, and zero. In particular, the overflow bit was dropped.

For our purposes these are not major deficiencies. The programming taught in the course is on a program segment basis, rather than a complete program basis.

Instruction Set

TORO instructions consist of an 8-bit opcode plus an 8-bit operand specification, except for inherent instructions (unary ALU operations) which have no operand. The operand specification may be a data value, an address, or a displacement value depending on the address mode.

The basic instruction consists of one of four operations on either of two registers, utilizing one of four address modes. If the operation is an ALU function or a branch (conditional), an additional function select specifies the ALU function or branch condition. These features are described in detail below. The 8-bit instruction format is given by

CCMMSSSR,

Opcode Format: CCMMSSSR

CC – OPERATION CLASS		MM – ADDRESS MODE		R – REGISTER SELECT	
00	LOAD	00	INHERENT	0	A
01	STORE	01	IMMEDIATE	1	X
10	BRANCH	10	DIRECT		
11	ALU	11	INDEXED		

SSS – FUNCTION SELECT

	ALU UNARY (INHERENT)	ALU BINARY	BRANCH MNEMONIC	BRANCH CONDITION
000	SHR	AND	JMP	1
001	SHL	OR	BEQ	Z
010	ROR	XOR	BNE	Z'
011	ROL	—	BLT	N
100	INC	ADD	BGE	N'
101	DEC	SUB	BGT	N'·Z'
110	COM	—	BCS	C
111	TST	CMP	BCC	C'

Fig. 1. Operation code definition.

where

$$CC = \text{Operation Class}$$
$$MM = \text{Address Mode}$$
$$SSS = \text{Function Select}$$
$$R = \text{Register Select.}$$

Fig. 1 gives complete operation code details. The four address modes (Inherent, Immediate, Direct, Indexed) used are representative of those found on real machines and are sufficient for covering hardware addressing techniques. The function select bits are used both for the ALU and branch instructions. Because inherent addressing signifies unary operations, the machine is capable of having 16 ALU functions. Of these, 14 have been used. The branch condition select has only eight options, with unconditional branch (jump) consuming one of them. With only one bit available for a register specification, the opcode can specify one of only two registers. They are the required accumulator and index register.

Fig. 2 gives the instruction set provided to students. Students are expected to be able to verify that the opcodes are correct by inspecting the values of each of the four fields. Not all of the opcodes that the machine will perform are presented to the student: many of the operations involving the index register are omitted, and only the branch immediate instructions (called branch direct by many vendors) are presented. This is done to conform with Motorola 6800 instructions. Of course, just because an opcode is missing from this set does not imply that it does not exist: the register transfer descriptions of the instructions presented later is the final authority on instruction definition.

Data Flow Design

The single-address architecture implies four registers—an accumulator (A), a program counter (PC), an instruction register (IR), and a memory address register (MAR). The indexed addressing feature necessitates an index register (X). For the purpose of making the data flow straightforward, the TORO uses a tristate-buffered bus as the primary data path component and edge-triggered registers for data storage. Fig. 3 gives the data flow diagram, along with the control points and control unit signals.

The ALU is treated as a black box that performs as previously described. Its output is loaded directly on the main data bus through tristate buffers. Because either the A or X register can be operated on by the ALU, a local multiplexed bus connects these two registers to one of the ALU data inputs. The select line on this eight-line 2-to-1 multiplexer is driven directly by the R (register select) bit in the IR, thus simplifying the control unit proper. The two registers A and X can actually be thought of as a single register R from the point of view of the control unit. A data path connects this local bus to the main data bus so that the control unit can load the data bus with the currently selected (A or X) register. A counterpart to this local bus is a demultiplexer controlled by the same bit of the IR. It routes the Load R control described later to the load control of either the A or X register.

The fact that the ALU directly loads the main data bus necessitates a temporary register (TMP) to hold the second operand for the ALU during binary operations. Additionally, a 3-bit status register (S) made up of a carry, a negative, and a zero bit is needed to store status information generated by the ALU. There are no other components in the data flow hardware.

Ten of the 14 data flow control points (shown in Fig. 3 as lines terminating in small circles) serve to enable the appropriate data value onto the bus or to load this value into the appropriate register (or memory). This aspect of the implementation is precisely what makes this computer so easy to understand. The control unit does little more than specify data movement—the equivalent of simple assignment statements in a high-level language. Most items that tend to complicate the control unit, such as specifying the ALU control bits or interpreting status bits, have been handled outside the control unit in a straightforward way.

The other four control points that are activated by the control unit are Load S (status), Incr PC, Clear T, and Index. The status register is only loaded during the execution of an ALU class instruction, a deviation from the way a 6800 (and other microprocessors) work. Clear T

	INHERENT		IMMEDIATE	DIRECT	INDEXED
LOAD					
LODA			10	20	30
LODX			11	21	31
STORE					
STOA				60	70
STOX				61	71
BRANCH					
JMP			90		
BEQ			92		
BNE			94		
BLT			96		
BGE			98		
BGT			9A		
BCS			9C		
BCC			9E		
ALU					
Unary		Binary			
SHRA	C0	ANDA	D0	E0	F0
SHLA	C2	ORA	D2	E2	F2
RORA	C4	XORA	D4	E4	F4
ROLA	C6				
INCA	C8	ADDA	D8	E8	F8
INCX	C9				
DECA	CA	SUBA	DA	EA	FA
DECX	CB				
COMA	CC				
TSTA	CE	CMPA	DE	EE	FE
		CMPX	DF	EF	FF

Fig. 2. TORO 680 instruction set.

controls the clear input of the timer register (T), a synchronous clear counter in the control unit. The decoded value from the timer register provides the sequencer information to the controller. The ability to clear the timer register allows instructions to use only the number of cycles they require.

Index is by far the most complicated of the control signals. It is required in order to signal the hardware that an indexed address needs to be computed. In an indexed instruction, preparatory signals first cause the offset value to be loaded into the TMP register in preparation for addition to the address contained in the X register. The indexed address is then calculated by the ALU, and the result loaded into the MAR.

The purpose of the Index control signal, then, is to route the X register to the ALU and cause the ALU to perform the ADD operation. These tasks are normally under the direct control of the IR. When Index is asserted, however, the values provided by the IR are temporarily overridden. A simple OR gate is used on the register select to force a value of one, which selects the X register. A similar scheme has been used to force the bits of the ALU function select to 100, the ADD code, when Index is active (see Fig. 3); when Index is not active, the IR function select bits are sent undisturbed to the ALU.

It is important that the student understand that all the registers are positive-edge-triggered devices. As such, the actions specified by the control unit during a given time cycle will occur at the beginning of the next (which can be thought of as occurring between cycles). The exception to this is the memory write. It requires only the combinational delay of the memory chip after the Write signal is asserted, not the next leading edge of the master clock. In a hardware implementation the Write signal could not be asserted until the last half of the master clock cycle to allow for address settling time.

Instruction Interpretation

The sequence of microoperations necessary to interpret (fetch and execute) each instruction is written using standard register transfer notation (RTN), as described in Langdon [9]. The instruction fetch is identical for all instructions and consumes cycles $T0$ and $T1$. For all address modes except inherent, an address calculation follows, in which the source or destination address for the operand is determined. The number of microoperations required for this phase depends on the address mode used. Finally, the instruction is executed.

The microoperation sequences for all operation class/address mode combinations are shown in Fig. 4. This figure highlights TORO design features that make it an attractive teaching machine. First, it shows the uniformity of the instruction set. The address calculation phase for a given address mode is independent of operation class, and the instruction execution phase for a given operation class is independent of address mode. Since the operation class and address mode constitute distinct fields in the opcode, it is easy for the student to make the connection between the instruction and the sequence of microoperations required to interpret it.

Fig. 4 also shows the symmetry between branch and load instructions. When the branch is to be taken (Br =

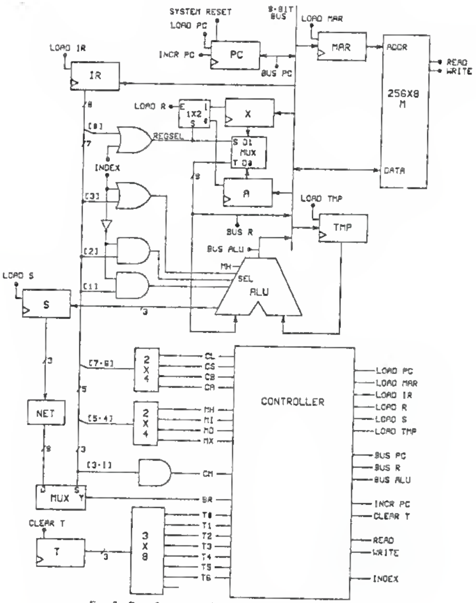

Fig. 3. Data flow, control points, and control unit signals.

1), the branch instruction is identical to the load instruction, with the exception that the PC, rather than A or X, is loaded. This symmetry provides sophisticated branch instructions (direct and indexed) while actually simplifying the control unit design.

Control Unit Design

Fig. 5 shows the inputs available to the control unit, and the outputs it needs to produce. Notable among the control unit inputs are the branch (Br) and compare (Cm) flags. If Br is asserted during a branch instruction, it informs the controller that the branch is to be taken; thus, the controller must assert Load PC to allow the address of the next instruction to be placed in the PC. The assertion of Cm during an ALU operation informs the controller that a compare or test instruction is executing. Compare and test are different from other ALU operations in that the selected register is not loaded with the result of the operation. Therefore, if Cm is asserted during an ALU operation, the controller must set Load R to 0.

The relationship between register transfer statement types and control points is listed in Fig. 6 (e.g., "$R \leftarrow$ anything" is associated with Load R). The conditions under which a given control point is asserted can then be identified by finding all occurrences of the associated RTN statements in Fig. 4. A product term is formed for each instance of a given RTN statement by ANDing all input variables listed in its row and column heading. Control point gathering is accomplished by collecting all such product terms for a given control point and ORing them together to form the unsimplified control unit equation for that control point. This process is repeated for all control points to produce the unsimplified control unit.

The simultaneous simplification of multiple Boolean equations is beyond the scope of the course, so the simplified control equations are presented without derivation. (The simplified control unit equations are given in Fig. 7.) Students are expected, however, to be able to verify that the simplified equations are correct.

The Simulator

The TORO simulator is a full-screen simulator for the IBM PC and compatibles that can run user programs in either single-instruction or single-cycle mode. In addition

```
                 Load Instructions
    Cl-Mh:                          Cl-Mi:
T0: Unused                      T0: MAR ← PC,       PC ← PC + 1
T1:   .                         T1: IR  ← M[MAR]
T2:   .                         T2: MAR ← PC,       PC ← PC + 1
T3:   .                         T3: T   ← 0,        R  ← M[MAR]

    Cl-Md:                          Cl-Mx:
T0: MAR ← PC,     PC ← PC + 1   T0: MAR ← PC,       PC ← PC + 1
T1: IR  ← M[MAR]                T1: IR  ← M[MAR]
T2: MAR ← PC,     PC ← PC + 1   T2: MAR ← PC,       PC ← PC + 1
T3: MAR ← M[MAR]                T3: TMP ← M[MAR]
T4: T   ← 0,      R  ← M[MAR]   T4: MAR ← X + TMP
T5:                             T5: T   ← 0,        R  ← M[MAR]

                 Store Instructions
    Cs-Mh:                          Cs-Mi:
T0: Unused                      T0: Unused
T1:   .                         T1:   .
T2:   .                         T2:   .
T3:   .                         T3:   .

    Cs-Md:                          Cs-Mx:
T0: MAR ← PC,     PC ← PC + 1   T0: MAR ← PC,       PC ← PC + 1
T1: IR  ← M[MAR]                T1: IR  ← M[MAR]
T2: MAR ← PC,     PC ← PC + 1   T2: MAR ← PC,       PC ← PC + 1
T3: MAR ← M[MAR]                T3: TMP ← M[MAR]
T4: T   ← 0,      M[MAR] ← R    T4: MAR ← X + TMP
T5:                             T5: T   ← 0,        M[MAR] ← R

                 Branch Instructions
    Cb-Mh:                          Cb-Mi:
T0: Unused                      T0: MAR ← PC,       PC ← PC + 1
T1:   .                         T1: IR  ← M[MAR]
T2:   .                         T2: MAR ← PC,       PC ← PC + 1
T3:   .                         T3: T   ← 0,     Br:PC ← M[MAR]

    Cb-Md:                          Cb-Mx:
T0: MAR ← PC,     PC ← PC + 1   T0: MAR ← PC,       PC ← PC + 1
T1: IR  ← M[MAR]                T1: IR  ← M[MAR]
T2: MAR ← PC,     PC ← PC + 1   T2: MAR ← PC,       PC ← PC + 1
T3: MAR ← M[MAR]                T3: TMP ← M[MAR]
T4: T   ← 0,   Br:PC ← M[MAR]   T4: MAR ← X + TMP
T5:                             T5: T   ← 0,     Br:PC ← M[MAR]

                 ALU Instructions
    Ca-Mh:                          Ca-Mi:
T0: MAR ← PC,    PC ← PC + 1    T0: MAR ← PC,       PC ← PC + 1
T1: IR  ← M[MAR]                T1: IR  ← M[MAR]
T2: T   ← 0,  Cm':R ← (op)R*    T2: MAR ← PC,       PC ← PC + 1
T3:                             T3: TMP ← M[MAR]
T4:                             T4: T   ← 0,  Cm':R ← R(op)TMP*

    Ca-Md:                          Ca-Mx:
T0: MAR ← PC,    PC ← PC + 1    T0: MAR ← PC,       PC ← PC + 1
T1: IR  ← M[MAR]                T1: IR  ← M[MAR]
T2: MAR ← PC,    PC ← PC + 1    T2: MAR ← PC,       PC ← PC + 1
T3: MAR ← M[MAR]                T3: TMP ← M[MAR]
T4: TMP ← M[MAR]                T4: MAR ← X + TMP
T5: T   ← 0,  Cm':R ← R(op)TMP* T5: TMP ← M[MAR]
T6:                             T6: T   ← 0,  Cm':R ← R(op)TMP*

* The status register S is assigned a value during this operation
  regardless of the value of Cm.
```

Fig. 4. Microoperation sequences.

to displaying the contents of the registers and memory, the TORO simulator also displays and animates the control unit equations.

Fig. 7 shows the simulator in operation. The registers are displayed in the upper left hand corner of the screen. The current bus value is displayed to the right of the registers, as are the control unit inputs, the mnemonic for the opcode that is currently in the instruction register, and the register select. Half of the addressable memory (locations 00–7*F*) is displayed in the right half of the screen. The memory location pointed to by the MAR is highlighted in reverse video. Only half of the available memory could be displayed due to lack of screen space, but 128 bytes is still more than adequate for student programs.

The control unit equations appear below the register display. The simulator derives much of its educational value from the animation of these equations. The active control unit outputs (those which have logic value "true") during each microcycle are highlighted in reverse video, along with the active control unit inputs. The simulator allows the student to see not only what control points are active, but the conditions that caused their activation. This is invaluable in the presentation of the control unit, which is the most difficult portion of the machine for students to comprehend.

The simulator has been carefully written to faithfully follow the hardware design. If an undefined opcode is executed it will perform the same as it would in a hardware implementation. The adventurous student can have fun with this feature.

Simulator Operation

The TORO simulator consists of a top level COMMAND interpreter and a number of utilities which can be invoked from the COMMAND prompt. Machine lan-

```
The Control Unit receives the following inputs:

4   operation classes (Ca - ALU; Cb - branch; Cl - load;
                        Cs - store)
4   address modes      (Mh - inherent; Mi - immediate;
                        Md - direct;   Mx - indexed)
7   timing signals     (T0 - T6)
1   branch condition,   Br (Br = 1 if branching is to
                            occur)
1   compare flag,       Cm (Cm = 1 for CMP or TST; Cm = 0
                            otherwise)

17  inputs


The Control Unit produces the following outputs:

6   register loads
1   PC increment
1   Timer register clear
3   bus enables
2   memory controls (Read and Write)
1   index flag (Index = 1 when an indexed address is to
               be computed)

14  outputs
```

Fig. 5. Control unit inputs and outputs

Control line	RTN statement for which control line value = 1
Load PC	PC ← M[MAR]
Load MAR	MAR ← anything
Load IR	IR ← anything
Load R	R ← anything NOTE: R means either A or X.
Load T	T ← anything
Load S	S ← anything
Incr PC	PC ← PC + 1
Clear T	T ← 0
Bus PC	anything ← PC
Bus R	anything ← R
Bus ALU	R ← (op)R; R ← R(op)TMP; MAR ← X + TMP
Read	anything ← M[MAR]
Write	M[MAR] ← anything
Index	MAR ← X + TMP

Fig. 6. Relationship between RTN statements and active control lines.

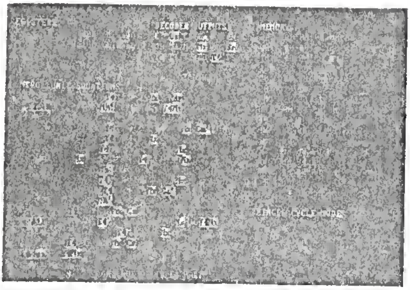

Fig. 7. The simulator during single-cycle program execution

guage programs can be entered via the PROGRAM command. The RUN command can then be used to execute the program. Programs can be run in either single-cycle or single-instruction mode, and can be HALTed at any time. SETx commands can be used to set the contents of any of the registers. Command descriptions are available through a HELP command.

In order to better understand the sort of information the simulator presents to the user, consider the following program, which fills four consecutive memory locations beyond location $6F$ with the value 54. The TORO Assembly language (utilizing simplified Motorola 6800 assembler syntax) for the program appears below:

```
        ORG   28      start program at address 28

        LODA  #54     value to be placed in memory
        LODX  #04     number of locations to be filled
FILL    STOA  6F,X    fill location beyond location 6F
        DECX          point to next location
        BNE   FILL    fill next location
HALT    JMP   HALT    stop program.
```

Fig. 7 shows the simulator during the execution of the program described above in single-cycle RUN mode. The screen display gives us a complete description of the internal state of the TORO processor. The decoder outputs show that it is currently time $T4$, and that the simulator is executing an instruction of operation class Cs (store) with address mode Mx (indexed). (This is line three of the program labeled FILL.) The MAR is pointing to the operand of the current instruction and the PC is pointing to the next instruction to be executed. The mnemonic display and IR contents indicate that the current operation is STOA, but RegSel = X. The latter tells us that Index is (temporarily) forcing the X register to be selected for an indexed address calculation. The branch flag Br is asserted, but since a store class instruction is being executed, this is a "DON'T CARE" condition. The active control unit equations are Load MAR, Bus ALU, and Index.

From the information supplied above, we can deduce what happens at time $T4$ of this instruction: the TORO computes an indexed address using the ALU (X + TMP = $6F$ + 02 = 71), transfers the result to the bus, then loads it into the MAR. On the next microcycle, the value in accumulator A will be stored at the address contained in the MAR. Note that although the address has been placed on the bus, it has yet to be placed in the MAR. This is in keeping with the timing constraint that register contents are altered only at the beginning of the next cycle.

Student Interaction with The Simulator

The TORO simulator has fulfilled its role admirably as a teaching aid. It has been employed both in classroom lectures and in one-on-one tutoring sessions in the instructor's office. But, most significantly, it has served as an exploratory tool for the individual student.

The TORO simulator was placed on a network of Zenith Z-150 computers in the Department of Computer Science at Kansas State University in the spring of 1986. The simulator was written in Basic, and compiled into a self-contained EXE file that the students can run on the networked machines or copy for use on other PC's. Students were assigned programming problems for the TORO which they hand-assembled and entered into the simulator's memory. Most of the programs were simple, requiring fewer than 20 bytes of machine code. Students were required to hand in their hand-assembled code, along with screen dumps of the simulator to show that the programs did indeed run. The TORO simulator was to be used ostensibly as a program debugging tool; however, it was hoped that students would also use the simulator to help them understand the basic operation of the TORO.

Student response to the simulator has been very positive. In a survey of 80 students conducted in the spring of 1986, 95 percent found the TORO simulator to be a good learning tool. Students commented that the "hands-on" experience of the simulator helped them not only to debug their assignments, but to gain a deeper understanding of the TORO itself. Students also found it enjoyable: they spent an average of four hours with the simulator, some of it "just playing." An additional indicator of the simulator's popularity was that despite its ready access on the network, 35 percent of the students surveyed said they either had copied or would copy the TORO simulator program for use on another computer.

Conclusion

The TORO machine and the TORO simulator constitute a valuable instructional resource for an introductory course in computer engineering. The TORO hardware is simple enough to be understood by the beginning student, but is complete enough to support the development of Assembly language concepts. The TORO thus bridges the gap found in introductory computer engineering textbooks between the simplistic machines used to teach hardware concepts and the complex machines used to teach programming.

The TORO simulator is a unique program that allows students to gain "hands-on" experience with the TORO. It serves both as an instructional tool and as a testbed for student programs. Its full-screen user interface not only displays register and memory contents, but also animates the control unit equations. The availability of single-cycle mode allows students to gain an in-depth understanding of the TORO machine.

References

[1] C. H. Roth, *Fundamentals of Logic Design*. St Paul, MN: West, 1985.

[2] M. M. Mano, *Digital Design*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[3] T. L. Booth, *Introduction to Computer Engineering: Hardware and Software Design*. New York: Wiley, 1984, ch. 12.

[4] M. M. Mano, *Digital Logic and Computer Design*. Englewood Cliffs, NJ: Prentice-Hall, 1979, ch. 11.

[5] F. J. Hill and G. R. Peterson, *Digital Logic and Microprocessors*. New York: Wiley, 1984, ch. 11.

[6] T. C. Bartee, *Digital Computer Fundamentals*. New York: McGraw-Hill, 1985, ch. 10.

[7] M. E. Sloan, *Computer Hardware and Organization*. Chicago, IL: Science Research Associates, 1983, ch. 11.
[8] *M6800 Programming Reference Manual*, Motorola Semiconductor Products, Phoenix, AZ, 1976.
[9] G. G. Langdon, Jr., *Computer Design*. San Jose, CA: Computeach, 1982, pp. 70-73.

John J. Devore (S'83-M'84) received the B.S. degree in physics, the M.S. degree in computer science, and the Ph.D. degree in electrical engineering from Kansas State University, Manhattan, in 1971, 1973, and 1984, respectively.

From 1970 to 1982 he was employed by the KSU Computing Center where he held such positions as System Programmer, Chief Consultant, and Manager of Programming Services. In 1984 he joined the Faculty of the Department of Electrical Engineering at KSU. His research interests include digital image processing, digital hardware design, computer algorithms, and cellular automata.

Dr. Devore is a member of Phi Kappa Phi and Tau Beta Pi Honorarys.

David S. Hardin (S'81) was born in Frankfort, KY, on December 7, 1960. He received the B.S. and M.S. degrees in electrical engineering from the University of Kentucky, Lexington, in 1982 and 1983, respectively.

He is currently a Ph.D. degree candidate in the Department of Electrical and Computer Engineering at Kansas State University, Manhattan, and a Technical Staff Member in the Collins Government Avionics Division of Rockwell International, Cedar Rapids, IA. His research interests include artificial intelligence, signal processing, and computer architecture.

Mr. Hardin is a member of Tau Beta Pi, Eta Kappa Nu, Phi Kappa Phi, the Association for Computing Machinery, and AAAI.

Store Instruction Modifications

Cs.Md

```
T0 :  MAR<--PC,     PC<--PC + 1
T1 :  IR <--M[MAR]
T2 :  MAR<--PC,     PC<--PC + 1
T3 :  MAR<--M[MAR]
T4 :  T  <--0,      WR<--R
T4':  T  <--0,      M[MAR]<--WR
T0 :
```

Cs.Mx

```
T0 :  MAR<--PC,     PC<--PC + 1
T1 :  IR <--M[MAR]
T2 :  MAR<--PC,     PC<--PC + 1
T3 :  TMP<--M[MAR]
T4 :  MAR<--X + TMP
T5 :  T  <--0,      WR<--R
T5':  T  <--0,      M[MAR]<--WR
T0 :
```

D Flip-Flop Data Input Equations For
The TORO 680-16 Four-Bit Program Counter Macro

$$D_0 = (D_A \cdot LD) + (Q_A' \cdot LD' \cdot EN) + (Q_A \cdot LD' \cdot EN')$$

$$D_1 = (D_B \cdot LD) + (Q_B' \cdot Q_A \cdot LD' \cdot EN) + (Q_B \cdot LD' \cdot Q_A') + (Q_B \cdot LD' \cdot EN')$$

$$D_2 = (D_C \cdot LD) + (Q_C' \cdot Q_B \cdot Q_A \cdot LD' \cdot EN) + (Q_C \cdot LD' \cdot EN') + (Q_C \cdot Q_B' \cdot LD') + (Q_C \cdot Q_A' \cdot LD')$$

$$D_3 = (D_D \cdot LD) + (Q_D' \cdot Q_C \cdot Q_B \cdot Q_A \cdot LD' \cdot EN) + (Q_D \cdot LD' \cdot EN') + (Q_D \cdot Q_C' \cdot LD') + (Q_D \cdot Q_B' \cdot LD') + (Q_D \cdot Q_A' \cdot LD')$$

$$RCO = Q_A \cdot Q_B \cdot Q_C \cdot Q_D \cdot EN$$

Where:

D_A, D_B, D_C, & D_D are the data load inputs,

Q_A, Q_B, Q_C, & Q_D are the counter outputs,

LD is the load enable input,

EN is the count enable input,

RCO is the ripple carry out.

D Flip-Flop Data Input Equations For The TORO 630-16 T Phase Clock

$$D_0 = (CLR' . Q_0')$$

$$D_1 = (CLR' . Q_1' . Q_0) + (CLR' . Q_1 . Q_0')$$

$$D_2 = (CLR' . Q_2' . Q_1 . Q_0) + (CLR' . Q_2 . Q_1') + (CLR' . Q_2 . Q_0')$$

Where:

D_0, D_1, & D_2 are the D flip-flop data inputs,

Q_0, Q_1, & Q_2 are the T phase clock outputs,

CLR is the synchronous reset enable.

THE VLSI DESIGN OF A SIMPLE-INSTRUCTION 16-BIT MICROPROCESSOR

by

JOSEPH EUGENE VARRIENTOS

BSEE, Kansas State University, 1986

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department Of Electrical And Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

The VLSI Design Of A Simple-Instruction 16-Bit Microprocessor

The research involved the VLSI layout and verification of a microprocessor originally designed as a teaching tool. The paper design of the microprocessor is currently used to introduce assembly language programming to students in Electrical Engineering and Computer Science, and hardware designs are constructed by students using discrete integrated logic circuits. It is the purpose of this research to clearly define functional boundaries of the design, add features to increase computing power, and to include the design in a single package so that it may be easily interfaced with existing microprocessor peripheral integrated circuits.

This 15,000-transistor CMOS design was accomplished on a SUN 3/60 workstation with the use of layout design tools from the Microelectronics Center of North Carolina and the Northwest Laboratory For Integrated Systems. The layout was constructed with design rules and processing parameters for the 3-micron, bulk p-well, scalable CMOS fabrication process available from MOSIS at the University of Southern California. The first set of tools was used to design a standard cell library, to simulate propagation delays, and compute the power consumption of the final design. The second set of tools was used to floorplan and assemble standard cells into large functional blocks for the final design, and to assemble the final design into the pad frame in preparation for device fabrication.

THE VLSI DESIGN OF A SIMPLE-INSTRUCTION 16-BIT MICROPROCESSOR

by

JOSEPH EUGENE VARRIENTOS

BSEE, Kansas State University, 1986

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department Of Electrical And Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

The VLSI Design Of A Simple-Instruction 16-Bit Microprocessor

The research involved the VLSI layout and verification of a microprocessor originally designed as a teaching tool. The paper design of the microprocessor is currently used to introduce assembly language programming to students in Electrical Engineering and Computer Science, and hardware designs are constructed by students using discrete integrated logic circuits. It is the purpose of this research to clearly define functional boundaries of the design, add features to increase computing power, and to include the design in a single package so that it may be easily interfaced with existing microprocessor peripheral integrated circuits.

This 15,000-transistor CMOS design was accomplished on a SUN 3/60 workstation with the use of layout design tools from the Microelectronics Center of North Carolina and the Northwest Laboratory For Integrated Systems. The layout was constructed with design rules and processing parameters for the 3-micron, bulk p-well, scalable CMOS fabrication process available from MOSIS at the University of Southern California. The first set of tools was used to design a standard cell library, to simulate propagation delays, and compute the power consumption of the final design. The second set of tools was used to floorplan and assemble standard cells into large functional blocks for the final design, and to assemble the final design into the pad frame in preparation for device fabrication.